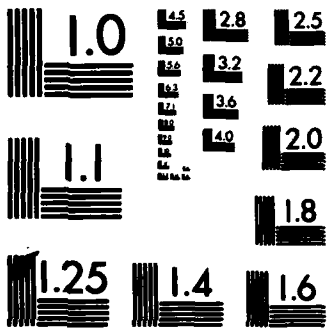
```
AD-A138 453    COMMAND GENERATOR TRACKER SYNTHESIS METHODS USING AN      1/4
               LQG (LINEAR SYSTEM M..(U) AIR FORCE INST OF TECH
               WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI..  J P MCMILLIAN
UNCLASSIFIED   DEC 83 AFIT/GE/EE/83D-46                  F/G 9/2     NL
```


1.0
4.5
5.0
5.6
6.3
7.1
8.0
9.0
10
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

COMMAND GENERATOR TRACKER SYNTHESIS METHODS USING AN LQG-DERIVED PROPORTIONAL PLUS INTEGRAL CONTROLLER BASED ON THE INTEGRAL OF THE REGULATION ERROR

THESIS

AFIT/GE/EE/83D-46 James P. McMillian
1st Lt USAF

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/EE/83D-46

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

DTIC COPY INSPECTED 2

COMMAND GENERATOR TRACKER SYNTHESIS METHODS USING AN LQG-DERIVED PROPORTIONAL PLUS INTEGRAL CONTROLLER BASED ON THE INTEGRAL OF THE REGULATION ERROR

THESIS

AFIT/GE/EE/83D-46 James P. McMillian
1st Lt USAF

AFIT/GE/EE/83D-46

COMMAND GENERATOR TRACKER SYNTHESIS METHODS USING AN LQG-DERIVED PROPORTIONAL PLUS INTEGRAL CONTROLLER BASED ON THE INTEGRAL OF THE REGULATION ERROR

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

by

James P. McMillian, B.S.

1st Lt USAF

Graduate Electrical Engineering

December 1983

Acknowledgments

I wish to express my deepest appreciation to Professor Peter S. Maybeck for his guidance throughout the duration of this work. I would also like to thank Captain Richard M. Floyd, whose invaluable experience and assistance made possible the successful completion of this project. Finally, I wish to extend a warm thank you to my wife Susan and daughters, JoAnne, Stefanie, and Jamie for their understanding and patience during the past many months.

Contents

List of Figures

List of Tables

AFIT/GE/EE/83D-46

Abstract

This study develops a computer program for interactive execution to aid in the design of Command Generator Tracker control systems employing Proportional Plus Integral inner loop controllers based on the integral of the regulation error and Kalman Filters for state estimation (CGT/PI/KF controllers). Sampled-data controller designs are based upon the Linear system model, Quadratic cost, and Gaussian noise process (LQG) assumptions of optimal control theory.

The report develops the CGT/PI/KF controller theory with the PI controller portion based upon the integral of the regulation error. Following a brief description of the computer program developed, results of applying it to an example aircraft-related controller design problem is presented. The CGT/PI/KF controller is found to be a technique particularly well suited to the aircraft control design problem but is a technique that may be applied to any general controller problem fitting the design criteria.

Use of the computer program is fully documented in the appendices of the report. Included are a brief "Programmer's Manual," a complete "User's Manual", and a program listing. These pertain to the computer program as implemented on a CDC CYBER computer system.

COMMAND GENERATOR TRACKER SYNTHESIS METHODS USING AN LQG-DERIVED PROPORTIONAL PLUS INTEGRAL CONTROLLER BASED ON THE INTEGRAL OF THE REGULATION ERROR

I. Introduction

1.1 Background

The earliest flight control designs served exclusively as a pilot relief function. It was much later before control systems were designed to improve aircraft stability, but these still had limited control authority. Furthermore, such systems were designed in a trial and error evolutionary fashion.

Current and future aircraft designs require increasingly complex controls. These newer aircraft employ digital flight control systems in order to meet complex performance specifications. The typical aircraft control design problem is one in which measurement data from many sensors are input, and signals to multiple control surfaces and actuators are output. These multiple-input/multiple-output control systems are required to respond to commanded inputs while simultaneously rejecting disturbances.

There are typically two general approaches to solving the problems of flight control. The oldest and most widely used and accepted is the classical design technique. The other technique is the "modern control" design. Despite the systematic nature and performance potential of the modern control design technique, this method has not supplanted, or even complemented to the degree originally anticipated, the classical control methodology. This is due in part to numerous deficiencies in initial applications of modern control design techniques. These deficiencies include: (1) traditional design criteria were not readily specified directly in the performance index, (2) it was difficult to select appropriate weighting matrices for the cost functionals to ensure desired response, (3) the resulting controllers required full-state feedback, but direct perfect measurements of all states are never available--thus requiring approximations or insertion of filters or observers before implementation, and (4) the original formulation of the optimal controller solved only the regulator problem and not the required tracking response to the command inputs.

To make modern control design techniques more applicable, a synthesis method known as model following, including implicit and explicit model following, has been used. Recent developments in modern control theory have been unified into a new synthesis technique known as command generator tracking (CGT)(Refs 2; 3; 4; and 18:151-166). In this methodology, one requires a system to respond to command inputs while rejecting disturbances, so that the states of the system maintain desired trajectories in real time. Both the desired trajectories and the disturbances are formulated as the outputs of linear system models. The first of these, the command generator model, can produce a time history of variables that the actual system is to follow or mimic, thereby incorporating handling qualities into the controller design (Refs 10 and 16).

In an open loop format, the problem is to determine appropriate gains, from the states of both the command generator model and the disturbance model or shaping filter (Ref 17:180-186) to the system control inputs (also to determine appropriate inputs to those models), that will yield command generator tracking; i.e. replication by the actual system of the desired output responses despite any modelled disturbances. If the original system is unstable or marginally stable, or affected by unmodelled disturbances and uncertainties, the addition of stabilizing feedback is required. The feedback controller should produce a system with a type-1 property to ensure the system will respond with zero steady-state error to a step input, since a more complex input can be approximated by a series of step inputs. Thus the consideration of a proportional plus integral (PI) design is appropriate for the feedback controller, since it can provide this type-1 property (Refs 5; 6; and 18:132-151). PI design based on augmenting the original state equations with difference equations for control variables, treating control pseudorates as driving functions, has already been exploited (Refs 9; 19; and 21). It has also been shown that analogous PI controllers can be obtained by augmenting the original system states with the integrals of regulation error (Ref 18:135-141). Finally, since perfect access to all system state variables is rarely available, sensor outputs must be processed by a Kalman filter (KF) or observer to generate estimates of the state variables (Ref 17:203-406). The resulting controller is known as a CGT/PI/KF controller.

To date, two efficient interactive computer software programs have been developed for the design and performance analysis of CGT/PI/KF controllers. The first software program, known as "CGTPIF" developed by Capt. Floyd (Ref 9) and later modified by Lt. Moseley (Ref 21), specifically aids the user in conducting the synthesis and analysis of the components of the CGT/PI/KF controller. The second, "PFEVAL" developed by Lt. Moseley, conducts a statistical performance analysis of the resulting composite controllers (Ref 21). "CGTPIF" currently bases the design of the PI controller only on the difference equations for control variables, but since some designers may be more familiar with, or have a better feel for, designing the PI controller based on the integral of the regulation error, the need for this type analysis tool exists.

1.2 Problem and Scope

The primary objectives of this thesis are:

1. To develop an interactive, user-oriented computer program similar to "CGTPIF", which will also interface with "PFEVAL", to aid in the design of CGT/PI/KF controllers, basing the design of the PI controller on the integral of the regulation error.

2. To apply the design program to an aircraft flight control problem to evaluate characteristics of this CGT/PI/KF controller design and to compare the designs.

1.3 Sequence of Presentation

The results of this effort are fully documented in the body of the thesis and in the appendices. Chapter II contains a short overview of the existing work performed in the previous two theses (Refs 9 and 21), with the theoretical aspects of the alternate PI controller design techniques presented in Chapter III. Chapter IV presents a description of the computer program developed for the alternate PI controller designs as well as summarizing the pertinent information from the existing "CGTPIF" program. Chapter V discusses and compares the PI controller designs through use of the program. A final chapter offers conclusions and recommendations for further research.

There are four appendices to this thesis. The first is a brief description of the computer program subroutines that were changed in going from "CGTPIF" to the alternate program called "CGTPIQ". For a complete program analysis, the reader of this thesis must also read Floyd's and Moseley's theses (Refs 9 and 21), especially if the plan is to understand all the computer program subroutines with the idea of modifying the program in any way. The second is a complete user's guide for "CGTPIQ". The third is a full source listing of the computer code and the fourth is a summary of the CGT/PI/KF design evaluation for CGTPIQ.

II. EXISTING CGT/PI/KF Design

2.1 Introduction

The control philosophy employed in the CGT/PI/KF controller is not new. While it is clearly in the class of model-following controllers and has characteristics closely related to those typical of the earlier model-following designs, the theory from which it is derived is distinctly not in a linear path with earlier work. In one consistent development it incorporates all the capabilities of earlier model-following designs, provides new capabilities, and does so in a single unified controller/filter structure.

The basic structure of this controller is as depicted in Fig. II-1. It accepts command inputs and generates a feedforward control through the command generator model and CGT controller; it incorporates a PI controller as an inner loop feedback controller to drive the system to follow the CGT inputs; and it uses a Kalman filter to provide estimates of both the system and disturbance states needed by the controllers.

Before presenting the theoretical development of the alternate PI controller (Chapter III), the rest of this chapter will introduce the CGT/PI/KF controller concept by briefly summarizing what is currently implemented in the interactive computer programs "CGTPIF" and "PFEVAL". This chapter is taken in part from a paper by Maybeck, Floyd, and

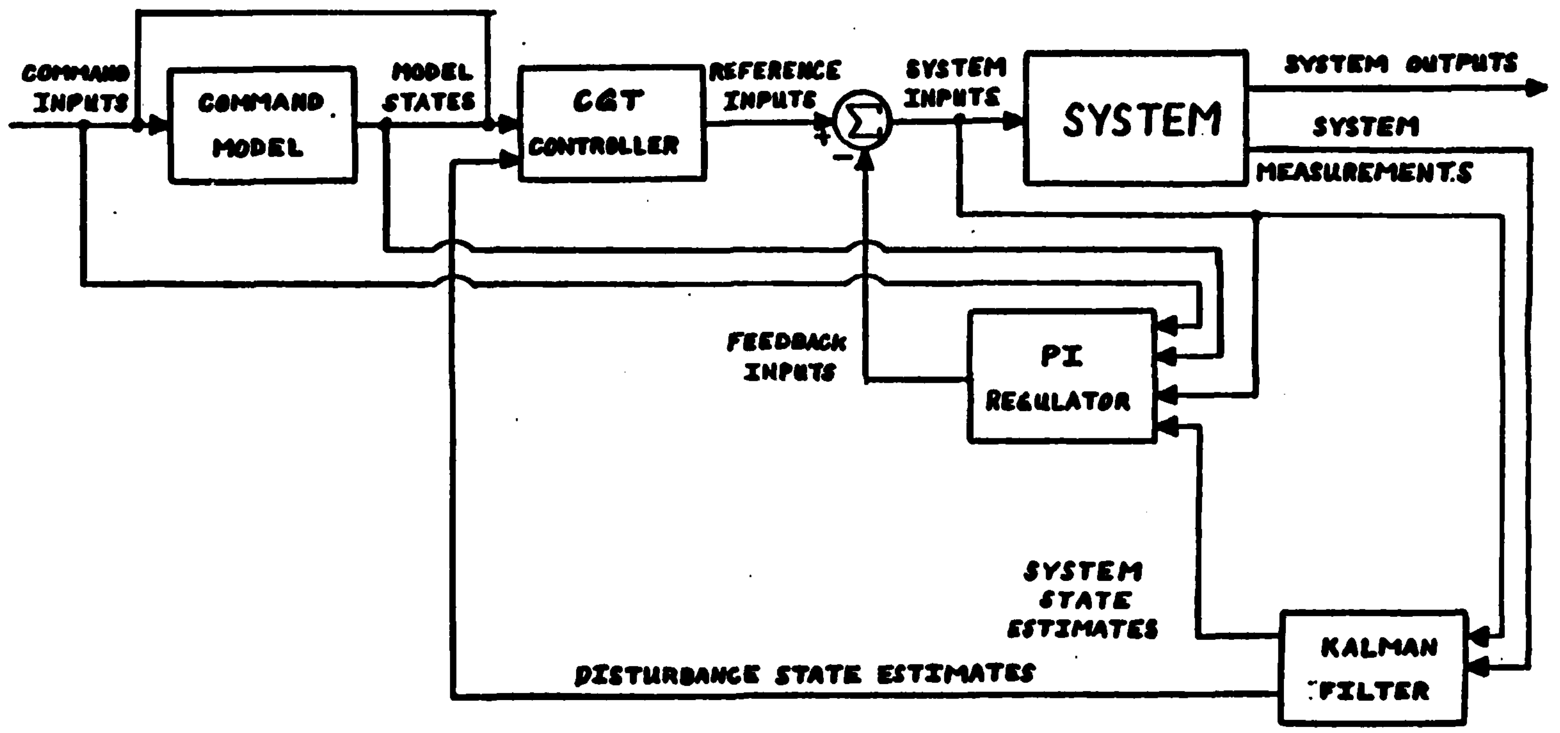


Fig. II-1. CGT/PI/KF General Block Diagram

Moseley (Ref 19) which presents this concept in a tutorial manner. The development presented in Chapter III will be constructed in a linear path with "CGTPIF" and will interface directly with "PFEVAL". It is recommended that the reader of this thesis also read the preceding theses (Refs 9 and 21) to obtain a more thorough overall picture of the problem.

2.2 Command Generator Tracking

Consider a system described by the linear time invariant discrete time model

$$\underline{x}(t_{i+1}) = \underline{\underline{\Phi}}\,\underline{x}(t_i) + \underline{\underline{B}}_d\underline{u}(t_i) + \underline{\underline{E}}_x\underline{n}_d(t_i) + \underline{w}(t_i) \quad \text{(II-1)}$$

$$\underline{y}_c(t_i) = \underline{\underline{C}}\,\underline{x}(t_i) + \underline{\underline{D}}_y\underline{u}(t_i) + \underline{\underline{E}}_y\underline{n}_d(t_i) \quad \text{(II-2)}$$

where $\underline{x}(t_i)$ is the state at sample time t_i, $\underline{u}(t_i)$ is the control applied to the system at time t_i, $\underline{n}_d(t_i)$ is a vector of time correlated noises and disturbances, $\underline{w}(t_i)$ is zero-mean white Gaussian dynamics driving noise of covariance $\underline{\underline{Q}}$, and $\underline{y}_c(t_i)$ is a vector of output controlled variables over which we want to exert influence and to cause to behave in a desirable manner. Such a model can be established by writing the continuous time linear perturbation equations for a system (such as an aircraft) and then generating the equivalent discrete time model at a chosen sample rate (Refs 17 and 18). This model is assumed to be stabilizable and detectable (Ref 18). In Equations (II-1) and (II-2)

$\underline{n}_d(t_i)$ is a vector of disturbances whose effects on the system we specifically want to reject, and it is modelled as the output of a shaping filter of the form

$$\underline{n}_d(t_{i+1}) = \underline{\Phi}_n \underline{n}_d(t_i) + \underline{G}_n \underline{w}_{dn}(t_i) \qquad \text{(II-3)}$$

where $\underline{w}_{dn}(t_i)$ is a zero-mean white Gaussian noise of covariance $\underline{Q}_n$ that is independent of $\underline{w}(t_i)$ in Equation (II-1). It is desired that the system described by these equations duplicate as closely as possible the output of a command generator model

$$\underline{x}_m(t_{i+1}) = \underline{\Phi}_m \underline{x}_m(t_i) + \underline{B}_m \underline{u}_m(t_i) \qquad \text{(II-4)}$$

$$\underline{y}_m(t_i) = \underline{C}_m \underline{x}_m(t_i) + \underline{D}_m \underline{u}_m(t_i) \qquad \text{(II-5)}$$

where the model input $\underline{u}_m(t_i)$ can be viewed as the desired input (such as provided by the pilot stick), and this model specifies the system's desired dynamic characteristics. Although $\underline{x}_m(t_i)$ and $\underline{u}_m(t_i)$ need not be of the same dimension as $\underline{x}(t_i)$ and $\underline{u}(t_i)$ of Equation (II-1), $\underline{y}_m(t_i)$ is of the same dimension as $\underline{y}_c(t_i)$ of Equation (II-2). For controller development, it is assumed that $\underline{u}_m(t_i)$ is a constant, $\underline{u}_m$. In actual applications it typically varies very slowly relative to the chosen sample period, and $\underline{u}_m$ can thus be considered piecewise constant with long intervals of essentially constant value. Since $\underline{u}_m$ is generally nonzero, a PI controller will be more desirable than a simple regulator for feedback, to enable achieving the desired values with zero steady state mean error. This will be pursued in the next section.

Mathematically, the objective of the CGT is to force the error between the actual system output, $\underline{y}_c$, and the desired (model) output, $\underline{y}_m$, to zero:

$$\underline{e}(t_i) = \underline{y}_c(t_i) - \underline{y}_m(t_i) = \underline{0} \qquad \text{(II-6)}$$

Assume initially that $\underline{x}(t_i)$, $\underline{n}_d(t_i)$, and $\underline{y}_c(t_i)$ are all perfectly accessible, to be replaced later by Kalman filter estimates. Thus, the ideal state and control trajectories, $\underline{x}_I(t_i)$ and $\underline{u}_I(t_i)$ for all t_i, are defined as the time histories the system states and controls must follow so that the true system output perfectly matches the model output, as in Equation (II-6); while the system is being affected by modelled disturbances:

$$\underline{x}_I(t_{i+1}) = \underline{\Phi}\underline{x}_I(t_i) + \underline{B}\underline{u}_I(t_i) + \underline{E}_x\underline{n}_d(t_i) \qquad \text{(II-7)}$$

Another requirement levied on the ideal trajectories is that they must be linear functions of $\underline{x}_m(t_i)$, $\underline{u}_m(t_i)$ and $\underline{n}_d(t_i)$:

$$\begin{bmatrix} \underline{x}_I(t_i) \\ \underline{u}_I(t_i) \end{bmatrix} = \begin{bmatrix} \underline{A}_{11} & \underline{A}_{12} & \underline{A}_{13} \\ \underline{A}_{21} & \underline{A}_{22} & \underline{A}_{23} \end{bmatrix} \begin{bmatrix} \underline{x}_m(t_i) \\ \underline{u}_m(t_i) \\ \underline{n}_d(t_i) \end{bmatrix} \qquad \text{(II-8)}$$

The CGT problem is then to solve for the matrix partitions in Equation (II-8) such that Equations (II-6), (II-7), and (II-8) are all satisfied. This will be shown in Chapter III (Section 3.3.1). Once these evaluations are accomplished, the CGT control law is implemented simply as the lower partition of Equation (II-8):

$$\underline{u}_I(t_i) = \underline{A}_{21}\underline{x}_m(t_i) + \underline{A}_{22}\underline{u}_m(t_i) + \underline{A}_{23}\underline{n}_d(t_i) \qquad \text{(II-9)}$$

2.3 PI Controller

Now consider the PI controller of Fig. II-1. One seeks a controller structure that includes accepting the sensed error between $\underline{y}_m(t_i)$ and achieved $\underline{y}_c(t_i)$, and generating the appropriate control to feed back into the system to keep this difference small. To keep the system in a nonzero equilibrium condition such that steady state error is zero, the controller fed by the regulation error $[\underline{y}_m(t_i) - \underline{y}_c(t_i)]$ must be able to deliver the appropriate nonzero steady state control when its own input is zero. This cannot be achieved by a simple proportional feedback regulator, and so a PI controller is motivated. This form of controller will also be able to reject the effect of unmodelled constant disturbances (as due to generating controller designs based on linear perturbation models expanded about a particular point in the operational envelope of the vehicle, while the vehicle is actually operating at a somewhat different point), which is also highly desirable in this application.

In this section, perfect access to all states and pertinent variables is again assumed, and the Linear system, Quadratic cost, Gaussian noise (LQG) synthesis technique will be used. This methodology provides a systematic synthesis procedure to generate a controller with desired stability properties ensured, in which the appropriate cross feeds in this multiple input/multiple output controller (whose evaluation may not be very apparent from other design techniques) are dictated by system and cost descriptions, and in which tradeoffs between state and control amplitudes can be readily accomplished in the design process.

2.3.1 PI Controller Based on Control Difference Equations

One means of developing a PI controller for this application is to consider the perturbation variables

$$\delta\underline{x}(t_i) = \underline{x}(t_i) - \underline{x}_I(t_i) \quad \text{(II-10a)}$$

$$\delta\underline{u}(t_i) = \underline{u}(t_i) - \underline{u}_I(t_i) \quad \text{(II-10b)}$$

$$\delta\underline{y}_c(t_i) = \underline{y}_c(t_i) - \underline{y}_m(t_i) \quad \text{(II-10c)}$$

In terms of these variables, perturbations about the ideal trajectories can be described by the augmented perturbation state equation

$$\begin{bmatrix} \delta\underline{x}(t_{i+1}) \\ \delta\underline{u}(t_{i+1}) \end{bmatrix} = \begin{bmatrix} \underline{\Phi} & \underline{B}_d \\ \underline{0} & \underline{I} \end{bmatrix} \begin{bmatrix} \delta\underline{x}(t_i) \\ \delta\underline{u}(t_i) \end{bmatrix} + \begin{bmatrix} \underline{0} \\ \underline{I} \end{bmatrix} \Delta\underline{u}(t_i) \quad \text{(II-11)}$$

where the upper partition is the perturbation equation associated with the system and the lower partition treats $\delta \underline{u}(t_i)$ as additional states by considering them as the output of a summation (pseudointegration) process

$$\delta \underline{u}(t_{i+1}) = \delta \underline{u}(t_i) + \Delta \underline{u}(t_i) \qquad \text{(II-12)}$$

in terms of control differences or "pseudorates". For this augmented perturbation system description, an optimal constant gain controller to minimize the quadratic cost

$$J = \sum_{i=0}^{\infty} \begin{bmatrix} \delta \underline{x}(t_i) \\ \delta \underline{u}(t_i) \\ \Delta \underline{u}(t_i) \end{bmatrix}^T \begin{bmatrix} \underline{X}_{11} & \underline{X}_{12} & \underline{S}_1 \\ \underline{X}_{12}^T & \underline{X}_{22} & \underline{S}_2 \\ \underline{S}_1^T & \underline{S}_2^T & \underline{U} \end{bmatrix} \begin{bmatrix} \delta \underline{x}(t_i) \\ \delta \underline{u}(t_i) \\ \Delta \underline{u}(t_i) \end{bmatrix} \qquad \text{(II-13)}$$

can be efficiently generated via solution to steady state (algebraic) Riccati difference equations (Ref 13). $\underline{X}_{11}$ specifies the cost weighting on state magnitudes, $\underline{X}_{22}$ on control magnitudes, and $\underline{U}$ on control differences. The solution form for this controller problem is

$$\Delta \underline{u}^*(t_i) = - [\underline{G}_{c1}^* \quad \underline{G}_{c2}^*] \begin{bmatrix} \delta \underline{x}(t_i) \\ \delta \underline{u}(t_i) \end{bmatrix} \qquad \text{(II-14)}$$

In terms of the controller gains $\underline{G}_{c1}^*$ and $\underline{G}_{c2}^*$, the final incremental PI control law can be written as (Ref 18:145)

$$\underline{u}(t_i) = \underline{u}(t_{i-1}) - \underline{K}_x[\underline{x}(t_i) - \underline{x}(t_{i-1})]$$
$$+\underline{K}_z[\underline{y}_m(t_i) - \underline{y}_c(t_{i-1})] \tag{II-15}$$

where $\underline{K}_x$ and $\underline{K}_z$ are given by

$$\underline{K}_x = \underline{G}_{c1}^{*}\underline{\Pi}_{11} + \underline{G}_{c2}^{*}\underline{\Pi}_{21} \tag{II-16a}$$
$$\underline{K}_z = \underline{G}_{c1}^{*}\underline{\Pi}_{12} + \underline{G}_{c2}^{*}\underline{\Pi}_{22} \tag{II-16b}$$

with

$$\begin{bmatrix} \underline{\Pi}_{11} & \underline{\Pi}_{12} \\ \underline{\Pi}_{21} & \underline{\Pi}_{22} \end{bmatrix} = \begin{bmatrix} (\underline{\Phi}-\underline{I}) & \underline{B}_d \\ \underline{C} & \underline{D}_y \end{bmatrix}^{-1} \tag{II-17}$$

which will be developed further in Chapter III.

2.3.2 Closed Loop CGT/PI Controller Based on Control Difference Equations

Without going into any detail as to its development, the incremental closed loop CGT/PI controller equation based on the control difference (as currently implemented in "CGTPIF") will be shown here. After reading Chapter III, the reader should refer back to this section to see the similarities in the two CGT/PI control laws. The law is

$$\begin{aligned}
\underline{u}(t_i) = \underline{u}(t_{i-1}) &- \underline{K}_x[x(t_i) - \underline{x}(t_{i-1})] \\
&+ \underline{K}_z\left\{[\underline{C}_m \quad \underline{D}_m]\begin{bmatrix}\underline{x}_m(t_{i-1}) \\ \underline{u}_m(t_i)\end{bmatrix} - [\underline{C} \quad \underline{D}_y]\begin{bmatrix}\underline{x}(t_{i-1}) \\ \underline{u}(t_{i-1})\end{bmatrix}\right\} \\
&+ [\underline{K}_x\underline{A}_{11} + \underline{A}_{21}][\underline{x}_m(t_i) - \underline{x}_m(t_{i-1})] \\
&+ [\underline{K}_x\underline{A}_{12} + \underline{A}_{22}][\underline{u}_m(t_i) - \underline{u}_m(t_{i-1})] \\
&+ [\underline{K}_x\underline{A}_{13} + \underline{A}_{23}][\underline{n}_d(t_i) - \underline{n}_d(t_{i-1})] \qquad \text{(II-18)}
\end{aligned}$$

Issues such as speedup and initial conditions will be addressed in Chapter III and are applicable to Equation (II-18). (The full development is presented in Refs 9 and 18:135-147).

2.4 Kalman Filter

The previous sections developed controllers under assumption that perfect access to all states is available at each sample time. Since this is generally not the case, a Kalman Filter is used to generate optimal estimates of these states from the incomplete noise corrupted outputs of physical sensors. Certainty equivalence (Ref 18) under the LQG assumptions yields the optimal controller as the controllers presented previously, but with $\underline{x}(t_i)$ and $\underline{n}_d(t_i)$ replaced by the estimates provided by the filter.

The dynamics model upon which the filter is based is generated by augmenting Equations (II-1) and (II-3) to get

$$\begin{bmatrix} \underline{x}(t_{i+1}) \\ \underline{n}_d(t_{i+1}) \end{bmatrix} = \begin{bmatrix} \underline{\Phi} & \underline{E}_x \\ \underline{0} & \underline{\Phi}_n \end{bmatrix} \begin{bmatrix} \underline{x}(t_i) \\ \underline{n}_d(t_i) \end{bmatrix} + \begin{bmatrix} \underline{B} \\ \underline{0} \end{bmatrix} \underline{u}(t_i) + \begin{bmatrix} \underline{I} & \underline{0} \\ \underline{0} & \underline{G}_n \end{bmatrix} \begin{bmatrix} \underline{w}(t_i) \\ \underline{w}_n(t_i) \end{bmatrix} \tag{II-19}$$

and the measurement devices are modelled by

$$\underline{z}(t_i) = [\underline{H} \quad \underline{H}_n] \begin{bmatrix} \underline{x}(t_i) \\ \underline{n}_d(t_i) \end{bmatrix} + \underline{v}(t_i) \tag{II-20}$$

where $\underline{v}$ is zero mean white Gaussian noise of covariance $\underline{R}$, independent of $\underline{w}$ and $\underline{w}_n$. From these models, a standard Kalman filter (or observer) can be derived and implemented in the constant-gain steady state form to accept measurements $\underline{z}(t_i)$ and produce $\hat{\underline{x}}(t_i^+)$ and $\hat{\underline{n}}_d(t_i^+)$, as shown in Fig. II-1. In the CGT/PI control law, $\underline{y}_c(t_i)$ is replaced by $\hat{\underline{y}}_c(t_i^+)$ in Equation (II-16) where from Equation (II-2)

$$\hat{\underline{y}}_c(t_i^+) = \underline{C}\hat{\underline{x}}(t_i^+) + \underline{D}_y\underline{u}(t_i) + \underline{E}_y\hat{\underline{n}}_d(t_i^+) \tag{II-21}$$

2.4.1 CGT/PI/KF Control Law Based on Control Difference Equations

The Kalman filter outputs can be combined with the CGT/PI control law to produce a CGT/PI/KF control law. The development of this law is thoroughly documented in Lt. Moseley's thesis as part of "PFEVAL" and is

$$\begin{aligned}\underline{u}(t_i) = &- \underline{K}_x(I-KH)\hat{\underline{x}}(t_i^-) - \underline{K}\underline{H}_n\hat{\underline{n}}_d(t_i^-) + \underline{K}\underline{z}(t_i)\\ &+ [\underline{K}_x\underline{A}_{11} + \underline{A}_{21}]\underline{x}_m(t_i)\\ &+ [\underline{K}_x\underline{A}_{13} + \underline{A}_{23}][-\underline{K}_n\underline{H}\hat{\underline{x}}(t_i^-) + (\underline{I}-\underline{K}_n\underline{H}_n)\hat{\underline{n}}_d(t_i^-)]\\ &+ \underline{K}_n\underline{z}(t_i) + \underline{K}_f\underline{f}(t_i)\end{aligned} \tag{II-22a}$$

where

$$\underline{K}_f(t_i) = \underline{K}_f(t_{i-1}) + [\underline{C}_m \quad \underline{D}_m]\begin{bmatrix}\underline{x}_m(t_{i-1})\\ \underline{u}_m(t_i)\end{bmatrix} - [\underline{C} \quad \underline{D}_y]\begin{bmatrix}\underline{x}(t_{i-1})\\ \underline{u}(t_{i-1})\end{bmatrix} \tag{II-22b}$$

and where the Kalman filter gains are given by

$$\begin{bmatrix}\underline{K}\\ \underline{K}_n\end{bmatrix} = \begin{bmatrix}\underline{P}_{xx} & \underline{P}_{xn}\\ \underline{P}_{xn}^T & \underline{P}_{nn}\end{bmatrix}\begin{bmatrix}\underline{H}^T\\ \underline{H}_n^T\end{bmatrix}\left\{[\underline{H} \quad \underline{H}_n]\begin{bmatrix}\underline{P}_{xx} & \underline{P}_{xn}\\ \underline{P}_{xn}^T & \underline{P}_{nn}\end{bmatrix}\begin{bmatrix}\underline{H}^T\\ \underline{H}_n^T\end{bmatrix} + \underline{R}\right\}^{-1} \tag{II-23}$$

2.4.2 Implicit and Explicit Model Following Based on Control Difference Equations

A modification performed by Lt. Moseley and documented in his thesis was to add implicit model following to the explicit model following of the "CGTPIF" program. In implicit model following, the model is incorporated into the performance index and thus helps determine appropriate feedback gains so that deviations in achieved transient response from the desired model response are penalized mathematically, and thus minimized operationally. In explicit model following, the model is used not only to determine appropriate feedback gains, but also feedforward gains on the model states themselves, thereby requiring an explicit simulatlon of the model dynamics in the controller itself. (For a thorough explanation of implicit and explicit model refer to Capt. Floyd's thesis (Ref 9), which very adequately summarizes the available literature on the subject.)

The result of adding implicit model following was to change the weighting on the $\underline{x}$ vector, with $\underline{x}$ defined as in Equation (II-11) as

$$\underline{x} = \begin{bmatrix} \int \underline{x} \\ \int \underline{u} \end{bmatrix} \tag{II-24}$$

from weights

$$\underline{X} = \begin{bmatrix} \underline{X}_{11} & \underline{X}_{12} \\ \underline{X}_{12}^T & \underline{X}_{22} \end{bmatrix} \tag{II-25}$$

as in Equation (II-13), to

$$\underline{X} = \begin{bmatrix} \underline{X}_{11} + \hat{\underline{X}}_I & \underline{X}_{12} + \hat{\underline{S}}_I \\ \underline{X}_{12}^T + \hat{\underline{S}}_I^T & \underline{X}_{22} + \hat{\underline{U}}_I \end{bmatrix} \tag{II-26}$$

where (Ref 18:68-82)

$$\underline{X}_{11} = \underline{C}^T \underline{Y} \underline{C} \tag{II-27a}$$

$$\underline{X}_{22} = \underline{U}_y + \underline{D}_y{}^T\underline{Y}\underline{D}_y \tag{II-27b}$$

$$\underline{X}_{12} = \underline{C}^T\underline{Y}\underline{D}_y \tag{II-27c}$$

in terms of a quadratic weighting matrix $\underline{Y}$ on $\delta\underline{y}_c$ and a weighting matrix $\underline{U}_y$ on $\delta\underline{u}$ in an original cost function definition convenient for a designer's use, and (Ref 9 and 21)

$$\hat{\underline{X}}_I = (\underline{CA} - \underline{A}_m\underline{C})^T\underline{X}_I(\underline{CA} - \underline{A}_m\underline{C}) \tag{II-28a}$$

$$\hat{\underline{S}}_I{}^T = \underline{B}^T\underline{C}^T\underline{X}_I(\underline{CA} - \underline{A}_m\underline{C}) \tag{II-28b}$$

$$\hat{\underline{U}}_I = \underline{U}_I + \underline{B}^T\underline{C}^T\underline{X}_I\underline{CB} \tag{II-28c}$$

(These same relations for $\underline{X}_{11}$, $\underline{X}_{22}$, $\underline{X}_{12}$, $\hat{\underline{X}}_I$, $\hat{\underline{S}}_I$, $\hat{\underline{U}}_I$ will appear in Chapter III for the alternate derivation of the PI controller based on the integral of the regulation error but the placement within the overall matrix will change. The reader may wish to refer back to this section after reading Chapter III.)

2.5 Performance Evaluation

It is desired to use the previous systematic controller design method for realistic applications in which online computer limitations such as speed and memory size are important constraints on the design. Thus, the system models used for controller design are purposely reduced order, reduced complexity models. A number of prospective CGT/PI/KF controllers can be developed, differing in state dimension, choice of states, complexity of defining matrices, performance index weighting matrices, and filter tuning parameters.

For these CGT/PI/KF controllers, or any other digital controllers, it is desirable to conduct a performance analysis as depicted in Fig. II-2. The truth model portrays the actual system to be controlled and the effects of the external environment upon the system, as well as can be modelled, regardless of complexity or computational loading. Therefore, the dimension of the truth model state $\underline{x}_t$ is typically much higher than that of the models used for controller design. For the most useful form of performance evaluation algorithm, one desires the ability to specify a continuous-time description of the truth model, and a discrete-time description of the digital controller with a sample frequency that is a variable design parameter. For performance evaluation purposes, it is important to portray the characteristics of the truth model state $\underline{x}_t$ and the commanded controls $\underline{u}$, as shown as outputs in Fig. II-2. It is the truth model state, depicting actual system characteristics, and not the state of the model that the controller assumes to be an adequate description, that is at issue. Furthermore, it is desired to evaluate the level of commanded controls, as to verify that they do not exceed certain physical limits or design specifications.

In a stochastic model setting, the mean and rms time histories of these variables could be generated via Monte Carlo analysis in the most general case. However, if the

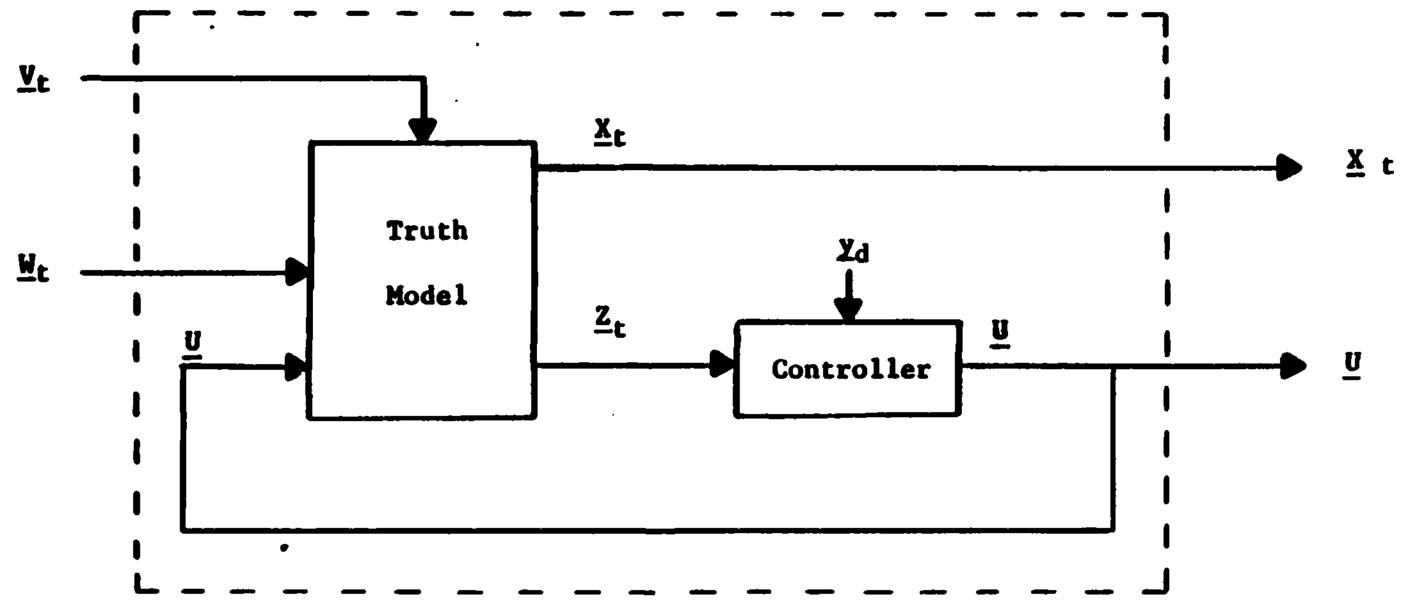


Fig. II-2. Performance Evaluation of Sampled Data Controller

truth model is itself linear, then such statistical time histories can be calculated explicitly, without requiring a Monte Carlo analysis of many simulation runs. This is what is accomplished in "PFEVAL".

2.6 Software Tools

Two efficient interactive computer software tools have been developed for the design and performance analysis of CGT/PI/KF controllers (Refs 9 and 21). Although one very important application is for the design of advanced digital flight controllers, these tools are in fact general purpose and can address a wide variety of control design problems.

The first software tool, known as "CGTPIF", specifically aids the user in conducting the synthesis of the components of the CGT/PI/KF controller and some analysis of characteristics of each of these components, as discussed in Sections 2.2-2.4. The second, "PFEVAL", conducts a statistical performance analysis of the resulting composite controllers, as discussed in Section 2.5. Some of their useful attributes for a user-friendly environment are:

1) each executes interactively;
2) each utilizes efficient array allocation;
3) various modes of entry are possible for the dynamics models for the design, command generator, and truth systems;
4) prespecified design paths are automatically followed, with user prompts at necessary decision points;
5) requests for inputs include informative prompts;
6) copious program output is provided; and

7) error checking is performed, and messages are given as appropriate.

Because of a significant usage of matrix manipulations, these programs employ an efficient library of routines (Ref 13). Aside from this library, "CGTPIF" contains about 2900 lines of source code and "PFEVAL" has about 1800. For ease of use, they are:

1) written in ANSI Standard Fortran IV;
2) highly portable;
3) implemented on a Control Data Corp. CYBER machine; and
4) segmented to achieve the necessary load size for interactive use without impacting the source code.

"CGTPIF" has three basic design paths: (1) design of a PI controller (using explicit and/or implicit quadratic weights), (2) design of either an open loop CGT or closed loop CGT/PI controller, and (3) design of a Kalman filter. As each of these components is generated, its performance is automatically evaluated: either controller path is automatically followed by deterministic analyses such as pole placement computations and step response plots (against a system described by either the design model or a truth model) while the filter generation is automatically followed by a covariance analysis of filter estimation accuracy in a truth model environment.

Initially, the user establishes the continuous time design model upon which to base the controller and filter, and also the desired sample period (the software generates all needed discretized models). Any of the three design paths can then be chosen, typically in the numerical order as listed previously. If the PI controller design is pursued prior to the CGT design path, then the subsequent CGT design will automatically be of the CGT/PI controller. If the PI is not yet determined during the current execution of the program, the designer may elect to design either a CGT or a CGT/PI controller. The CGT design is not pursued if the open loop design model is unstable. When any of these design paths are completed, the user is given the opportunity to loop on the design path, to choose a different design path, or to terminate program execution. Each traversal of a design path yields a proposed component for use in Fig. II-1, and all pertinent information is saved in a file for processing by "PFEVAL".

"PFEVAL" specifically combines proposed CGT/PI and Kalman filter designs (as mentioned in Section 2.4.1) to generate an entire composite controller as in Fig. II-1, and then subjects these composite controllers to a covariance performance analysis of the closed loop system, as shown in Fig. II-2. Immediately after each such CGT/PI/KF controller is evaluated, the corresponding full state feedback CGT/PI is also tested against the same truth model, specifically to indicate the impact of the filter on closed loop system properties. By repeatedly testing the same controller against varied truth models, the important impact of the filter on robustness (Ref 18:102-114) can be portrayed explicitly.

Together, the two tools described provide a user-friendly environment for efficient and systematic design and performance evaluation of CGT/PI/KF controllers. Applicability of the design approach and software tools to the following objectives has been demonstrated (Refs 9 and 21):

1) design of a conventional pitch rate controller for a modern fighter aircraft (AFTI/F-16);

2) decoupled pitch pointing control for the same aircraft;

3) effect of design model order reduction on closed loop system performance;

4) effect of parameter variations in the truth model upon closed loop system performance, and the impact of the filter upon controller robustness;

5) incorporation of implicit model following as well as explicit model following;

6) handling slowly changing disturbances via PI controller action versus direct rejection of modelled disturbances; and

7) incorporation of additional rolloff into the controller to address the inadequacy of linearized system models at higher frequencies due to neglected modes and other causes.

2.7 Summary

This chapter has introduced the CGT/PI/KF controller concept by briefly summarizing what is currently implemented in the interactive computer programs "CGTPIF" and "PFEVAL". Chapter III will now present the theoretical development of the alternate PI controller.

III. PI and CGT/PI Development

3.1 Introduction

The CGT/PI/KF design presented in Chapter II is currently incorporated into the two software packages previously discussed. Since the purpose of this thesis effort is to design a computer program that is similar to "CGTPIF" but bases the PI controller design on the integral of the regulation error, this chapter will develop only the equations pertinent to this effort. Also, any material contained in the previous theses which is needed or which will add continuity of thought to this effort will be repeated and incorporated into this development. This will prevent the reader from having to read Floyd's and Moseley's theses (Refs 9 and 21) to understand this development fully.

3.2 PI Controller Based on the Integral of the Regulation Error

The design goal in employing a PI controller is to generate a feedback controller which will maintain the deterministic system output at a nonzero commanded value with zero mean steady state error, despite unmodeled constant disturbances which may also drive the system. Invoking the "Certainty Equivalence" property (Ref 18), let the deterministic system be described by the time-invariant model

$$\underline{x}(t_{i+1}) = \underline{\underline{\Phi}}\,\underline{x}(t_i) + \underline{\underline{B}}_d\underline{u}(t_i) + \underline{d}; \quad \underline{x}(t_o) = \text{given} \qquad \text{(III-1)}$$

$$\underline{y}_c(t_i) = \underline{C}\underline{x}(t_i) + \underline{D}_y\underline{u}(t_i) \qquad \text{(III-2)}$$

Therefore, it is desired to generate a controller that will maintain the vector of controlled variables $\underline{y}_c(t_i)$ at a nonzero desired value $\underline{y}_d$ despite unknown disturbance $\underline{d}$. This is known as achieving a "type-one" property (Ref 18). One seeks a controller structure that includes accepting a sensed error between desired $\underline{y}_d$ and the achieved $\underline{y}_c(t_i)$, and generating the appropriate control to keep this small. To keep the system at a nonzero equilibrium condition such that the steady-state error is zero, the controller fed by this regulation error signal $[\underline{y}_d - \underline{y}_c(t_i)]$ must be able to deliver the appropriate nonzero steady state control when its own input is zero. This motivates the integral action to form a proportional plus integral or PI controller, accomplished by introducing an additional set of dynamic variables, with defining difference equations, which are then augmented to the original states. For Linear Quadratic Gaussian design of the PI controller, these additional variables can correspond to either (1) the "pseudointegral", or summation, of the (negative) regulation error

$$\begin{aligned}\underline{q}(t_i) &= \underline{q}(t_o) + \sum_{j=0}^{i-1}[\underline{y}_c(t_j) - \underline{y}_d] \\ &= \underline{q}(t_{i-1}) + [\underline{y}_c(t_{i-1}) - \underline{y}_d] \qquad \text{(III-3)}\end{aligned}$$

or (2) the control difference $\Delta\underline{u}(t_i)$, such that

$$\underline{u}(t_{i+1}) = \underline{u}(t_i) + \Delta\underline{u}(t_i)$$

where $\Delta\underline{u}(t_i)$ can be interpreted as a desired control rate times the sample period

$$\Delta\underline{u}(t_i) \cong [\dot{\underline{u}}(t_i)\Delta t]$$

and thus its alternate name of control "pseudorate". The second method was the method developed by Floyd in his thesis and currently implemented in the program "CGTPIF", and thus will not be discussed further. The method to be developed in this thesis is method 1, the "pseudointegral" of the regulation error.

The PI controller may be formulated for implementation in either of two forms, the "position form" or the "incremental form". The position form represents the current input in its entirety and does so in terms of the total values of the feedback variables. In the incremental form only the change in control input from its previous value is computed, and it is in terms of changes in the values of the feedback variables since the preceding sample period. The incremental form for the controller has certain advantages over the position form, such as less concern about proper initialization of augmented states, and is the method to be used for final implementation (Ref 18).

The optimal PI controller is first developed from a discrete-time problem formulation. Subsequently, the technique for translating a continuous-time quadratic cost formulation, which is the formulation with which most designers will work, to the appropriate discrete-time cost function is demonstrated.

3.2.1 PI Mathematical Development

Following the development of Ref (18), define a "pseudointegral" state $\underline{q}(t_i)$ as in Equation (III-3) where

$$[\underline{y}_c(t_i) - \underline{y}_d] = [\underline{C}\underline{x}(t_i) + \underline{D}_y\underline{u}(t_i) - \underline{y}_d] \qquad \text{(III-4)}$$

is the (negative) regulation error at time t_i. The system is assumed to be in equilibrium before t_o, so $\underline{q}(t_o)$ is equal to the equilibrium value attained up to that time, $\underline{q}_{old}$, of value to be specified (which will be done later at the same time $\underline{q}_o$ is established). Thus, this state satisfies Equation (III-3). The augmented system description then becomes (after shifting time arguments on $\underline{q}$, combining Equation (III-2) and (III-3), and setting $\underline{d}=\underline{0}$ in Equation (III-1), since $\underline{d}$ is not to be modelled in the controller generation)

$$\begin{bmatrix} \underline{x}(t_{i+1}) \\ \underline{q}(t_{i+1}) \end{bmatrix} = \begin{bmatrix} \underline{\Phi} & \underline{0} \\ \underline{C} & \underline{I} \end{bmatrix} \begin{bmatrix} \underline{x}(t_i) \\ \underline{q}(t_i) \end{bmatrix} + \begin{bmatrix} \underline{B}_d \\ \underline{D}_y \end{bmatrix} \underline{u}(t_i) - \begin{bmatrix} \underline{0} \\ \underline{I} \end{bmatrix} \underline{y}_d \qquad \text{(III-5)}$$

for $t_i \geq t_o$.

To satisfy the design goal in employing the PI controller, that of maintaining $\underline{y}_c(t_i)$ at the desired setpoint $\underline{y}_d$, an equilibrium solution must be found to yield $\underline{x}(t_i)=\underline{x}_o$ for all t_i, such that $\underline{y}_c(t_i)=\underline{y}_d$. The nominal control $\underline{u}_o$ to hold the system at that equilibrium point is found as the solution to

$$\underline{x}_o = \underline{\Phi}\underline{x}_o + \underline{B}_d\underline{u}_o \qquad \text{(III-6)}$$

$$\underline{y}_d = \underline{C}\underline{x}_o + \underline{D}_y\underline{u}_o \qquad \text{(III-7)}$$

or, rearranging,

$$\begin{bmatrix} (\underline{\Phi}-\underline{I}) & \underline{B}_d \\ \underline{C} & \underline{D}_y \end{bmatrix} \begin{bmatrix} \underline{x}_o \\ \underline{u}_o \end{bmatrix} = \begin{bmatrix} \underline{0} \\ \underline{y}_d \end{bmatrix} \qquad \text{(III-8)}$$

If we assume that the composite matrix in Equation (III-8) is square and invertible, then for any given $\underline{y}_d$, $\underline{x}_o$ and $\underline{u}_o$ can be found. If we let

$$\begin{bmatrix} (\underline{\Phi}-\underline{I}) & \underline{B}_d \\ \underline{C} & \underline{D}_y \end{bmatrix}^{-1} = \begin{bmatrix} \underline{\Pi}_{11} & \underline{\Pi}_{12} \\ \underline{\Pi}_{21} & \underline{\Pi}_{22} \end{bmatrix} \qquad \text{(II-18)}$$

then

$$\begin{bmatrix} \underline{x}_o \\ \underline{u}_o \end{bmatrix} = \begin{bmatrix} (\underline{\Phi}-\underline{I}) & \underline{B}_d \\ \underline{C} & \underline{D}_y \end{bmatrix}^{-1} \begin{bmatrix} \underline{0} \\ \underline{y}_d \end{bmatrix} \qquad \text{(III-9a)}$$

$$= \begin{bmatrix} \underline{\Pi}_{11} & \underline{\Pi}_{12} \\ \underline{\Pi}_{21} & \underline{\Pi}_{22} \end{bmatrix} \begin{bmatrix} \underline{0} \\ \underline{y}_d \end{bmatrix} \qquad \text{(III-9b)}$$

or

$$\underline{x}_o = \underline{\Pi}_{12}\underline{y}_d \qquad \text{(III-10a)}$$

$$\underline{u}_o = \underline{\Pi}_{22}\underline{y}_d \qquad \text{(III-10b)}$$

If the composite matrix in Equation (III-8) is not square, a solution can be found using the left or right inverses (Ref 18:124), but this will not be pursued here.

Having found $\underline{x}_o$ and $\underline{u}_o$, perturbation variables can be defined as

$$\delta\underline{x}(t_i) = \underline{x}(t_i) - \underline{x}_o = \underline{x}(t_i) - \underline{\Pi}_{12}\underline{y}_d \qquad \text{(III-11a)}$$

$$\delta\underline{u}(t_i) = \underline{u}(t_i) - \underline{u}_o = \underline{u}(t_i) - \underline{\Pi}_{22}\underline{y}_d \qquad \text{(III-11b)}$$

$$\delta\underline{y}_c(t_i) = \underline{y}_c(t_i) - \underline{y}_d \qquad \text{(III-11c)}$$

$$\delta\underline{q}(t_i) = \underline{q}(t_i) - \underline{q}_o \qquad \text{(III-11d)}$$

where $\underline{q}_o$ is the new "pseudointegral" state steady state value, yet to be determined.

In terms of the perturbation variables, the augmented system perturbation model becomes (from Equation (III-5))

$$\begin{bmatrix} \delta\underline{x}(t_{i+1}) \\ \delta\underline{q}(t_{i+1}) \end{bmatrix} = \begin{bmatrix} \underline{\Phi} & \underline{0} \\ \underline{C} & \underline{I} \end{bmatrix} \begin{bmatrix} \delta\underline{x}(t_i) \\ \delta\underline{q}(t_i) \end{bmatrix} + \begin{bmatrix} \underline{B}_d \\ \underline{D}_y \end{bmatrix} \delta\underline{u}(t_i) \qquad \text{(III-12)}$$

For the optimal regulator the quadratic cost criterion to be minimized is

$$J = \sum_{i=0}^{N} \left\{ \frac{1}{2} \begin{bmatrix} \delta \underline{x}(t_i) \\ \delta \underline{q}(t_i) \\ \delta \underline{u}(t_i) \end{bmatrix}^T \begin{bmatrix} \underline{X}_{11} & \underline{X}_{12} & \underline{S}_1 \\ \underline{X}_{12}^T & \underline{X}_{22} & \underline{S}_2 \\ \underline{S}_1^T & \underline{S}_2^T & \underline{U} \end{bmatrix} \begin{bmatrix} \delta \underline{x}(t_i) \\ \delta \underline{q}(t_i) \\ \delta \underline{u}(t_i) \end{bmatrix} \right\}$$
$$+ \begin{bmatrix} \delta \underline{x}(t_{N+1}) \\ \delta \underline{q}(t_{N+1}) \end{bmatrix}^T \begin{bmatrix} \underline{X}_{f11} & \underline{X}_{f12} \\ \underline{X}_{f12}^T & \underline{X}_{f22} \end{bmatrix} \begin{bmatrix} \delta \underline{x}(t_{N+1}) \\ \delta \underline{q}(t_{N+1}) \end{bmatrix} \qquad \text{(III-13)}$$

where $\underline{X}_{11}$ weights state deviation from the nominal $\underline{x}_o$, $\underline{X}_{22}$ weights deviations from the nominal "pseudointegral" state $\underline{q}_o$, and $\underline{U}$ weights control deviations from the nominal $\underline{u}_o$. The weights $\underline{X}_f$ apply to the deviations at terminal time, and for an infinite time duration steady state controller problem (i.e., $N \rightarrow \infty$), these weights will not be used.

By letting $N \rightarrow \infty$ in Equation (III-18), and solving for the steady-state control law, it was shown (Ref 18:1-60) that the solution is of the form

$$\underline{u}^*(t_i) = -[\underline{G}_{c1}^* \quad \underline{G}_{c2}^*] \begin{bmatrix} \delta \underline{x}(t_i) \\ \delta \underline{q}(t_i) \end{bmatrix} \qquad \text{(III-14a)}$$

$$= - \underline{G}_{c1}^* \delta \underline{x}(t_i) - \underline{G}_{c2}^* \delta \underline{q}(t_i) \qquad \text{(III-14b)}$$

Substituting Equations (III-11a), (III-11b), and (III-11d) into Equation (III-14b) yields

$$[\underline{u}^*(t_i) - \underline{u}_o] = - \underline{G}_{c1}^* [\underline{x}(t_i) - \underline{x}_o] - \underline{G}_{c2}^* [\underline{q}(t_i) - \underline{q}_o] \qquad \text{(III-15)}$$

or after multiplying and rearranging terms

$$\underline{u}^*(t_i) = - \underline{G}_{c1}{}^*\underline{x}(t_i) - \underline{G}_{c2}{}^*\underline{q}(t_i) + [\underline{u}_o + \underline{G}_{c1}{}^*\underline{x}_o + \underline{G}_{c2}{}^*\underline{q}_o] \quad \text{(III-16)}$$

with $\underline{q}(t_i)$ determined as in Equation (III-3).
$\underline{G}_{c1}{}^*$ and $\underline{G}_{c2}{}^*$ are found by considering the general form for $\underline{G}_c{}^*$ in Equation (III-14a) (Ref 18:73)

$$\underline{G}_c{}^* = [\underline{U} + \underline{B}_d{}^T\underline{K}_c\underline{B}_d]^{-1}[\underline{B}_d{}^T\underline{K}_c\underline{\Phi} + \underline{S}^T] \quad \text{(III-17)}$$

where $\underline{K}_c$ satisfies the backward Riccati equation

$$\underline{K}_c(t_i) = \underline{X} + \underline{\Phi}^T\underline{K}_c(t_{i+1})\underline{\Phi} - [\underline{B}_d{}^T\underline{K}_c(t_{i+1})\underline{\Phi} + \underline{S}^T]^T\underline{G}_c{}^* \quad \text{(III-18a)}$$

solved backwards from the terminal condition

$$\underline{K}_c(t_{n+1}) = \underline{X}_f \quad \text{(III-18b)}$$

to its steady state value, or, equivalently, solving the (algebraic) steady state version of Equation (III-18a) for the constant $\underline{K}_c = \underline{K}_c(t_i) = \underline{K}_c(t_{i+1})$. If we take our augmented system descriptions, Equations (III-5) and (III-13), and partition them so that each segment matches an entry in the general solution we have

$$[\underline{G}_{c1}^{*} \quad \underline{G}_{c2}^{*}] = \left\{ \underline{U} + [\underline{B}_d^T \quad \underline{D}_y^T] \begin{bmatrix} \underline{K}_{c11} & \underline{K}_{c12} \\ \underline{K}_{c12}^T & \underline{K}_{c22} \end{bmatrix} \begin{bmatrix} \underline{B}_d \\ \underline{D}_y \end{bmatrix} \right\}^{-1} \left\{ [\underline{B}_d^T \quad \underline{D}_y^T] \begin{bmatrix} \underline{K}_{c11} & \underline{K}_{c12} \\ \underline{K}_{c12}^T & \underline{K}_{c22} \end{bmatrix} \begin{bmatrix} \underline{\Phi} & \underline{0} \\ \underline{C} & \underline{I} \end{bmatrix} + [\underline{S}_1^T \quad \underline{S}_2^T] \right\} \tag{III-19}$$

Multiplying this out yields

$$\begin{aligned} [\underline{G}_{c1}^{*} \quad \underline{G}_{c2}^{*}] = \{ & \underline{U} + \underline{B}_d^T \underline{K}_{c11} \underline{B}_d + \underline{D}_y^T \underline{K}_{c12} \underline{B}_d \\ & + \underline{B}_d^T \underline{K}_{c12} \underline{D}_y + \underline{D}_y^T \underline{K}_{c22} \underline{D}_y \}^{-1} \{ [\underline{B}_d^T \underline{K}_{c11} \underline{\Phi} \\ & + \underline{D}_y^T \underline{K}_{c12}^T \underline{\Phi} + \underline{B}_d^T \underline{K}_{c12} \underline{C} + \underline{D}_y^T \underline{K}_{c22} \underline{C} + \underline{S}_1^T] \, \vdots \\ & [\underline{B}_d^T \underline{K}_{c12} + \underline{D}_y^T \underline{K}_{c22} + \underline{S}_2^T] \} \end{aligned} \tag{III-20}$$

Doing the same thing for $\underline{K}_c(t_i)$ yields

$$\begin{aligned} \begin{bmatrix} \underline{K}_{c11} & \underline{K}_{c12} \\ \underline{K}_{c12}^T & \underline{K}_{c22} \end{bmatrix} = & \begin{bmatrix} \underline{X}_{11} & \underline{X}_{12} \\ \underline{X}_{12}^T & \underline{X}_{22} \end{bmatrix} + \begin{bmatrix} \underline{\Phi}^T & \underline{C}^T \\ \underline{0} & \underline{I} \end{bmatrix} \begin{bmatrix} \underline{K}_{c11} & \underline{K}_{c12} \\ \underline{K}_{c12}^T & \underline{K}_{c22} \end{bmatrix} \begin{bmatrix} \underline{\Phi} & \underline{0} \\ \underline{C} & \underline{I} \end{bmatrix} \\ & - \left\{ [\underline{B}_d^T \quad \underline{D}_y^T] \begin{bmatrix} \underline{K}_{c11} & \underline{K}_{c12} \\ \underline{K}_{c12}^T & \underline{K}_{c22} \end{bmatrix} \begin{bmatrix} \underline{\Phi} & \underline{0} \\ \underline{C} & \underline{I} \end{bmatrix} \right. \\ & \left. + [\underline{S}_1^T \quad \underline{S}_2^T] \right\}^T [\underline{G}_{c1}^{*} \quad \underline{G}_{c2}^{*}] \end{aligned} \tag{III-21}$$

and multiplying this out yields

$$\begin{aligned} \underline{K}_{c11}(t_i) = & \underline{X}_{11} + \underline{\Phi}^T \underline{K}_{c11} \underline{\Phi} + \underline{C}^T \underline{K}_{c12}^T \underline{\Phi} + \underline{\Phi}^T \underline{K}_{c12} \underline{C} \\ & + \underline{C}^T \underline{K}_{c22} \underline{C} - [\underline{B}_d^T \underline{K}_{c11} \underline{\Phi} + \underline{D}_y^T \underline{K}_{c12}^T \underline{\Phi} \\ & + \underline{B}_d^T \underline{K}_{c12} \underline{C} + \underline{D}_y^T \underline{K}_{c22} \underline{C} + \underline{S}_1^T] \underline{G}_{c1}^{*} \end{aligned} \tag{III-22a}$$

$$\underline{K}_{c22}(t_i) = \underline{X}_{22} + \underline{K}_{c22} - [\underline{B}_d^T \underline{K}_{c12} + \underline{D}_y^T \underline{K}_{c22} + \underline{S}_2^T]\underline{G}_{c2}^* \qquad \text{(III-22b)}$$

$$\underline{K}_{c12}(t_i) = \underline{X}_{12} + \underline{\Phi}^T \underline{K}_{c12} + \underline{C}^T \underline{K}_{c22} - [\underline{B}_d^T \underline{K}_{c11} \underline{\Phi} + \underline{D}_y^T \underline{K}_{c12} \underline{\Phi} + \underline{B}_d^T \underline{K}_{c12} \underline{C} + \underline{D}_y^T \underline{K}_{c22} \underline{C} + \underline{S}_1^T]\underline{G}_{c2}^* \qquad \text{(III-22c)}$$

(all K's on the right are at time t_{i+1})

Now comes the question of appropriately choosing $\underline{q}_o$ so as to yield the best possible transient performance from Equation (III-16). We seek the equilibrium value $\underline{q}_o$ that results in the lowest total cost to complete the process. It has been shown that this value of $\underline{q}_o$ is found by minimizing (Ref 18:136-137)

$$\text{Jmin} = \tfrac{1}{2}[\delta\underline{x}^T(t_o) \quad \delta\underline{q}^T(t_o)] \begin{bmatrix} \underline{K}_{c11} & \underline{K}_{c12} \\ \underline{K}_{c12}^T & \underline{K}_{c22} \end{bmatrix} \begin{bmatrix} \delta\underline{x}(t_o) \\ \delta\underline{q}(t_o) \end{bmatrix} \qquad \text{(III-23)}$$

with respect to $\delta\underline{q}(t_o)$:

$$\underline{0} = \frac{\partial \text{Jmin}}{\partial[\delta\underline{q}(t_o)]}^T = \underline{K}_{c22}\delta\underline{q}(t_o) + \underline{K}_{c12}^T \delta\underline{x}(t_o) \qquad \text{(III-24)}$$

Solving for $\delta\underline{q}(t_o)$ yields

$$\delta\underline{q}(t_o) = \underline{q}_{old} - \underline{q}_o = -\underline{K}_{c22}^{-1}\underline{K}_{c12}^T[\underline{x}_{old} - \underline{\Pi}_{12}\underline{y}_d]$$
$$= -\underline{K}_{c22}^{-1}\underline{K}_{c12}^T\underline{\Pi}_{12}[\underline{y}_{d\ old} - \underline{y}_d] \qquad \text{(III-25)}$$

or

$$\underline{q}_o = -\underline{K}_{c22}{}^{-1}\underline{K}_{c12}{}^{T}\underline{\Pi}_{12}\underline{y}_d \qquad \text{(III-26a)}$$

$$\underline{q}_{old} = -\underline{K}_{c22}{}^{-1}\underline{K}_{c12}{}^{T}\underline{\Pi}_{12}\underline{y}_{d\ old} \qquad \text{(III-26b)}$$

Substituting Equations (III-26b), (III-10a), and (III-10b) into Equation (III-16) yields the final control law in position form

$$\underline{u}^*(t_i) = -\underline{G}_{c1}{}^*\underline{x}(t_i) - \underline{G}_{c2}{}^*\underline{q}(t_i) + \underline{E}\underline{y}_d(t_i) \qquad \text{(III-27a)}$$

$$\begin{aligned}\underline{q}(t_{i+1}) &= \underline{q}(t_i) + [\underline{y}_c(t_i) - \underline{y}_d(t_i)] \\ &= \underline{q}(t_i) + [\underline{C}\underline{x}(t_i) + \underline{D}_y\underline{u}(t_i) - \underline{y}_d(t_i)]\end{aligned} \qquad \text{(III-27b)}$$

where

$$\underline{E} = [\underline{G}_{c1}{}^* - \underline{G}_{c2}{}^*\underline{K}_{c22}{}^{-1}\underline{K}_{c12}{}^{T}]\underline{\Pi}_{12} + \underline{\Pi}_{22} \qquad \text{(III-28a)}$$

$$\underline{q}(t_o) = \underline{q}_{old} \qquad \text{(III-28b)}$$

The position form control law as given by Equation (III-27) is diagrammed in Fig. III-1. The structure of this controller is seen to be composed of (1) full state feedback through $\underline{G}_{c1}{}^*$, (2) command feedforward through $\underline{E}$, and (3) "pseudointegration" of the regulation error in the forward path.

By writing Equation (III-27a) for both t_i and t_{i-1}, explicitly subtracting, rearranging and incorporating Equation (III-3), the incremental form of this

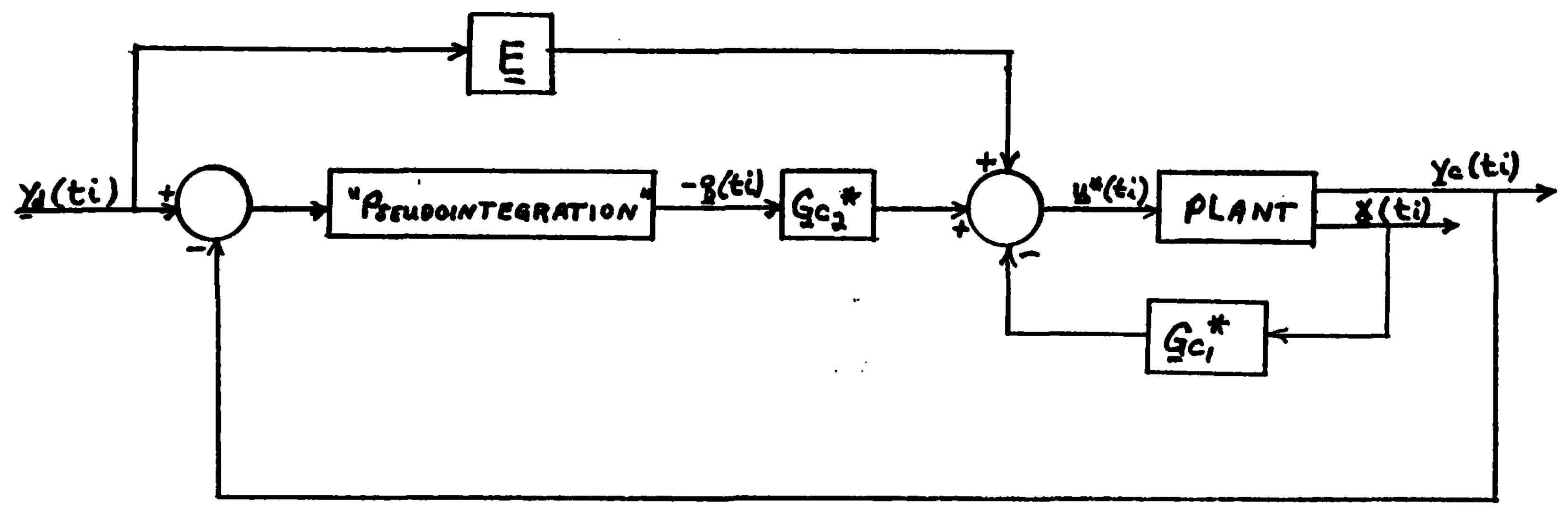


Fig. III-1. Position Form PI Control Law Based on the "Pseudointegration" of the Regulation Error

PI law is obtained as

$$\underline{u}^*(t_i) = \underline{u}^*(t_{i-1}) - \underline{G}_{c1}^*[\underline{x}(t_i) - \underline{x}(t_{i-1})]$$
$$+ \underline{G}_{c2}^*[\underline{y}_d(t_{i-1}) - \underline{y}_c(t_{i-1})]$$
$$+ \underline{E}[\underline{y}_d(t_i) - \underline{y}_d(t_{i-1})] \qquad \text{(III-29)}$$

There is no explicit "pseudointegral" state in this form, so we do not have to be concerned about proper initialization of augmented states as we do in position forms (Ref 18:139, 234-238). This is one reason why incremental forms are preferred (also see Ref 18:223-260). It has been shown that either form has the desired type-1 property if $\det\{\underline{G}_{c2}^*\} \neq \underline{0}$, and that $\underline{y}_c(t_i)$ converges to $\underline{y}_d$ in the limit as $t \to \infty$, despite the effect of unmodeled $\underline{d}$ (Ref 18:139). If we compare Equation (III-29) with Equation (II-15) we can see the similarities between the two PI controller developments. The similarities are pointed out in Ref (18:148-150) and summarized in the following discussion.

The two PI controllers can be interrelated by the generic form of the PI controller which is

$$\underline{u}^*(t_i) = -\underline{G}\underline{x}(t_i) + \underline{K}_p[\underline{y}_d(t_i) - \underline{y}_c(t_i)] + \underline{K}_i\underline{\xi}(t_i) \qquad \text{(III-30a)}$$

$$\underline{\xi}(t_{i+1}) = \underline{\xi}(t_i) + [\underline{y}_d(t_i) - \underline{y}_c(t_i)] \qquad \text{(III-30b)}$$

where the state feedback gain $\underline{G}$ and regulation error proportional gain $\underline{K}_p$ and integral gain $\underline{K}_i$ are given by

$$\underline{G} = (\underline{I} - \underline{E}\underline{D}_y)^{-1}(\underline{G}_{c1}{}^* - \underline{E}\underline{C}) \qquad \text{(III-31a)}$$
$$\underline{K}_p = (\underline{I} - \underline{E}\underline{D}_y)^{-1}\underline{E} \qquad \text{(III-31b)}$$
$$\underline{K}_i = (\underline{I} - \underline{E}\underline{D}_y)^{-1}\underline{G}_{c2}{}^* \qquad \text{(III-31c)}$$

for the PI controller based on the integral of the regulation error, and by

$$\underline{G} = (\underline{I} - \underline{K}_z\underline{D}_y)^{-1}(\underline{K}_x - \underline{K}_z\underline{C}) \qquad \text{(III-32a)}$$
$$\underline{K}_p = (\underline{I} - \underline{K}_z\underline{D}_y)^{-1}\underline{K}_z \qquad \text{(III-32b)}$$
$$\underline{K}_i = \underline{K}_p \qquad \text{(III-32c)}$$

for the PI controller based on control differences. Note that the proportional and integral gains are distinct in the PI controller based on the integral of the regulation error, whereas, they are the same in the PI controller based on control differences. This difference should allow the PI controller based on the integral of the regulation error to compensate better for applications in which the system is in fact nonlinear and linear techniques are being used to generate perturbation controls, i.e., for essentially all real applications of this synthesis technique. This should provide better compensation for an important phenomenon called windup (Ref 18).

3.2.2 Converting from Continuous Time Cost

Although the controller design determines a discrete-time control law, the design itself proceeds from continuous-time specifications. The system to be controlled is generally a continuous-time process and so it is appropriately defined by a continuous-time model.

Since the system to be controlled is a continuous-time system, and since its behavior is important at all time and not merely at the controller sample times, the cost function appropriate to the controller design is

$$J_c = \frac{1}{2}\int_{t_0}^{t_{N+1}} \begin{bmatrix} \underline{x}(t) \\ \underline{u}(t) \end{bmatrix}^T \begin{bmatrix} \underline{X}_c & \underline{S}_c \\ \underline{S}_c^T & \underline{U}_c \end{bmatrix} \begin{bmatrix} \underline{x}(t) \\ \underline{u}(t) \end{bmatrix} dt \qquad \text{(III-33)}$$

(plus a terminal quadratic involving $\underline{X}_f$ which is not used for an infinite time steady state controller problem, i.e. for $t_{N+1} \longrightarrow \infty$) and where

$$\underline{x} = \begin{bmatrix} \delta\underline{x} \\ \delta\underline{q} \end{bmatrix} \qquad \text{(III-34a)}$$

$$\underline{u} = \delta\underline{u} \qquad \text{(III-34b)}$$

$$\underline{X}_c = \begin{bmatrix} \underline{X}_{c11} & \underline{X}_{c12} \\ \underline{X}_{c12}^T & \underline{X}_{c22} \end{bmatrix} \qquad \text{(III-34c)}$$

Furthermore, since the design objective is to drive the system so that its output tracks the desired output, it is appropriate that the quadratic weighting matrices specified should apply to the system outputs, Equation (III-2) but on a continuous-time basis, and inputs, $\underline{u}$. Thus, defining the weights on output deviations as $\underline{Y}$, on input magnitudes as

$\underline{U}_y$, and on the "pseudointegral" state as $\underline{U}_q$, the components of the $\underline{X}_c$ matrix are obtained as (Refs 9 and 21)

$$\underline{X}_{c11} = \underline{C}^T\underline{Y}\underline{C} \qquad \text{(III-35a)}$$

$$\underline{X}_{c22} = \underline{U}_q \qquad \text{(III-35b)}$$

$$\underline{X}_{c12} = \underline{0} \qquad \text{(III-35c)}$$

$\underline{U}_c$ becomes

$$\underline{U}_c = \underline{U}_y + \underline{D}_y{}^T\underline{Y}\underline{D}_y \qquad \text{(III-36)}$$

and $\underline{S}_c$ becomes

$$\underline{S}_c = \begin{bmatrix} \underline{C}^T\underline{Y}\underline{D}_y \\ \underline{0} \end{bmatrix} \qquad \text{(III-37)}$$

where $\underline{C}$ and $\underline{D}_y$ are as defined in Equation (III-2) and $\underline{Y}$, $\underline{U}_y$ and $\underline{U}_q$ are positive semidefinite, positive definite and positive definite, respectively.

In order to use the continuous-time cost function of Equation (III-33) for solution of the discrete-time optimal controller, it is necessary to obtain the corresponding discrete-time cost function. Begin by conceptually dividing the control interval t_o to t_{N+1} into (N+1) control intervals of duration equal to the intended controller sample period T. The cost can then be expressed as

$$J = \sum_{i=0}^{N} \Big(\int_{t_i}^{t_i+T} \tfrac{1}{2}[\underline{x}^T(t)\underline{X}_C\underline{x}(t) + \underline{u}^T(t)\underline{U}_C\underline{u}(t) + 2\underline{x}^T(t)\underline{S}_C\underline{u}(t)]\,dt \Big) \tag{III-38}$$

where $\underline{u}(t)$ is assumed constant over a sampling period of T seconds and $\underline{x}(t)$ satisfies

$$\underline{x}(t) = \underline{\Phi}_f(t-t_i)\underline{x}(t_i) + \underline{B}_f(t-t_i)\underline{u}(t_i) \tag{III-39}$$

where for any argument τ

$$\underline{\Phi}_f(\tau) = \begin{bmatrix} \underline{\Phi}(\tau) & \underline{0} \\ \underline{C} & \underline{I} \end{bmatrix} \tag{III-40a}$$

with

$$\underline{\Phi}(\tau) = \exp\{\underline{A}\tau\} \tag{III-40b}$$

where $\underline{A}$ defines the homogeneous system dynamics in

$$\dot{\underline{x}} = \underline{A}\underline{x} + \underline{B}\underline{u} + \underline{G}\underline{w} \tag{III-40c}$$

Also,

$$\underline{B}_f(\tau) = \begin{bmatrix} \underline{B}_d(\tau) \\ \underline{D}_y \end{bmatrix} \tag{III-41a}$$

with

$$\underline{B}_d(\tau) = \int_0^{\tau} \underline{\Phi}(\sigma)\underline{B}\, d\sigma \tag{III-41b}$$

where $\underline{B}$ is as given in Equation (III-40c). From Equation (III-12), $\underline{\Phi}_\delta = \underline{\Phi}_\delta(t_{i+1}-t_i)$
and $\underline{B}_\delta = \underline{B}_\delta(t_{i+1}-t_i)$ are given by

$$\underline{\Phi}_\delta = \begin{bmatrix} \underline{\Phi} & \underline{0} \\ \underline{C} & \underline{I} \end{bmatrix} \tag{III-42a}$$

$$\underline{B}_\delta = \begin{bmatrix} \underline{B}_d \\ \underline{D}_y \end{bmatrix} \tag{III-42b}$$

The discrete time cost function then becomes

$$J = \sum_{i=0}^{N} \tfrac{1}{2} [\underline{x}^T(t_i)\underline{X}_\delta \underline{x}(t_i) + \underline{u}^T(t_i)\underline{U}_\delta \underline{u}(t_i) + 2\underline{x}^T(t_i)\underline{S}_\delta \underline{u}(t_i)] \tag{III-43a}$$

where

$$\underline{X}_\delta = \int_{t_i}^{t_i+T} [\underline{\Phi}_\delta{}^T(\tau)\underline{X}_c\underline{\Phi}_\delta(\tau)]\,d\tau \tag{III-43b}$$

$$\underline{U}_\delta = \int_{t_i}^{t_i+T} [\underline{B}_\delta{}^T(\tau)\underline{X}_c\underline{B}_\delta(\tau) + \underline{U}_c + \underline{B}_\delta{}^T(\tau)\underline{S}_c + \underline{S}_c{}^T\underline{B}_\delta(\tau)]\,d\tau \tag{III-43c}$$

$$\underline{S}_\delta = \int_{t_i}^{t_i+T} [\underline{\Phi}_\delta{}^T(\tau)\underline{X}_c\underline{B}_\delta(\tau) + \underline{\Phi}_\delta{}^T(\tau)\underline{S}_c]\,d\tau \tag{III-43d}$$

The integrals in Equations (III-43b,c,d) can be approximated in a two-step computation. First, $\underline{\Phi}_\delta$ and $\underline{B}_\delta$ are treated as constants over the sample interval with value set to their respective averaged values at the beginning and end of the interval:

$$\bar{\underline{\Phi}}_s = [\underline{\Phi}_s(0) + \underline{\Phi}_s(t_{i+1}-t_i)] = \tfrac{1}{2}[\underline{I} + \underline{\Phi}_s] \qquad \text{(III-44a)}$$
$$\bar{\underline{B}}_s = [\underline{B}_s(0) + \underline{B}_s(t_{i+1}-t_i)] = \tfrac{1}{2}[\underline{0} + \underline{B}_s] \qquad \text{(III-44b)}$$

(Where the overbar mean approximation). With these approximations, each of the integrands is constant over the integration time T, so the integrals can be obtained as

$$\bar{\underline{X}}_s = T[\bar{\underline{\Phi}}_s^T \underline{X} \bar{\underline{\Phi}}_s] \qquad \text{(III-45a)}$$
$$\bar{\underline{U}}_s = T[\bar{\underline{B}}_s{}^T \underline{X} \bar{\underline{B}}_s + \underline{U}_c + \bar{\underline{B}}_s{}^T \underline{S}_c + \underline{S}_c{}^T \bar{\underline{B}}_s] \qquad \text{(III-45b)}$$
$$\bar{\underline{S}}_s = T[\bar{\underline{\Phi}}_s^T \underline{X} \bar{\underline{B}}_s + \bar{\underline{\Phi}}_s^T \underline{S}_c] \qquad \text{(III-45c)}$$

where $T = (t_{i+1}-t_i)$ is the sample period. This provides a better approximate evaluation than simple Euler integration provides. Now, expressing this last cost function in matrix format becomes

$$J = \sum_{j=0}^{N} \frac{1}{2} \begin{bmatrix} \underline{x}(t_i) \\ \underline{u}(t_i) \end{bmatrix}^T \begin{bmatrix} \bar{\underline{X}}_s & \bar{\underline{S}}_s \\ \bar{\underline{S}}_s{}^T & \bar{\underline{U}}_s \end{bmatrix} \begin{bmatrix} \underline{x}(t_i) \\ \underline{u}(t_i) \end{bmatrix} \qquad \text{(III-46)}$$

Note that the cross weighting matrix $\bar{\underline{S}}_s$ has been introduced into the cost function by the discretization process and will generally be nonzero even if $\underline{S}_c$ in Equation (III-33) were zero. This cost function is seen to be in the standard form used previously and all development can proceed as if the problem were a discrete problem and not a continuous one.

Now, in order to obtain an equivalent discrete-time cost function with no cross weighting (to allow use of standard Riccati equation solvers as in (Ref 12) that assume such a form), define a new system (Ref 15)

$$\underline{x}(t_{i+1}) = \bar{\Phi}_s{}'\underline{x}(t_i) + \bar{B}_s\underline{u}'(t_i) \qquad \text{(III-47a)}$$

for which

$$\bar{\Phi}_s{}' = \bar{\Phi}_s - \bar{B}_s\bar{U}_s{}^{-1}\bar{S}_s{}^T \qquad \text{(III-47b)}$$

and

$$\underline{u}'(t_i) = \underline{u}(t_i) + \bar{U}_s{}^{-1}\bar{S}_s{}^T\underline{x}(t_i) \qquad \text{(III-47c)}$$

and for which the corresponding cost function to be minimized is

$$J' = \sum_{i=0}^{N} \frac{1}{2}\,[\underline{x}^T(t_i)\bar{X}_s{}'\underline{x}(t_i) + \underline{u}'^T(t_i)\bar{U}_s{}'\underline{u}'(t_i)] \qquad \text{(III-48a)}$$

with

$$\bar{X}_s{}' = \bar{X}_s - \bar{S}_s\bar{U}_s{}^{-1}\bar{S}_s{}^T \qquad \text{(III-48b)}$$

If the system of Equation (III-47a) is either controllable or stabilizable, letting $N \longrightarrow \infty$ leads to a steady state solution of the discrete Riccati equation represented as

(Ref 12)

$$\underline{K}_R' = \underline{\Phi}_s'^T \underline{K}_R' \underline{\Phi}_s' + \bar{\underline{X}}_s' \\ - \bar{\underline{B}}_s \underline{K}_R' \underline{\Phi}_s'^T [\bar{\underline{U}}_s + \bar{\underline{B}}_s^T \underline{K}_R' \bar{\underline{B}}_s]^{-1} \bar{\underline{B}}_s^T \underline{K}_R' \underline{\Phi}_s' \quad \text{(III-49)}$$

and the optimal feedback control is

$$\underline{u}^{*\prime}(t_i) = -\underline{G}_c^{*\prime} \underline{x}(t_i) \quad \text{(III-50)}$$

where

$$\underline{G}_c^{*\prime} = [\bar{\underline{U}}_s + \bar{\underline{B}}_s^T \underline{K}_R' \bar{\underline{B}}_s]^{-1} \bar{\underline{B}}_s^T \underline{K}_R' \underline{\Phi}_s' \quad \text{(III-51)}$$

The corresponding optimal feedback gain matrix for the original state system is

$$\underline{G}_c^* = \underline{G}_c^{*\prime} + \bar{\underline{U}}_s^{-1} \underline{S}_s^T \quad \text{(III-52)}$$

Remembering from Equation (III-19) that

$$\underline{G}_c^* = [\underline{G}_{c1}^* \quad \underline{G}_{c2}^*] \quad \text{(III-53)}$$

the calculation beginning with a continuous-time cost description will produce the correct $\underline{G}_c^*$. The only question that might arise is whether the substitution of $\underline{K}_R'$ for $\underline{K}_c$ is a valid substitution. (See Appendix A for derivation showing that $\underline{K}_R' = \underline{K}_c$). The $\underline{G}_c^*$ thus determined determines the appropriate gains to use in Equation (III-29) for the incremental PI control law.

3.3 CGT/PI Development

A natural extension of the controllers developed in the previous sections is provided by a Command Generator Tracker (CGT) in which we require a system to respond to command inputs, while rejecting disturbances, so that the states of the system maintain desired trajectories in real time. In an open loop conceptualization, the problem is to find the appropriate feedforward gains from the states of both the command generator and a disturbance linear model (and command generator inputs), to the system control inputs, that will yield command generator tracking. If the original system is unstable or just marginally stable, or if unmodeled disturbances and uncertainties affect the system, we are motivated to add the PI feedback as previously developed to the system.

The development presented in this section will start with the open loop format, developing the appropriate gains, and then closing the loop by adding the PI controller and utilizing the open loop gains to develop the complete CGT/PI controller. There are four formulations for this CGT/PI controller. Two are mathematically derived and two are ad-hoc. These four developments will be presented (all four CGT/PI controller formulations will be implemented in code (see Appendix C), and a comparison made between them (see Chapter V)).

3.3.1 Open Loop Command Generator Tracker

Consider a system described by the linear time invariant model

$$\underline{x}(t_{i+1}) = \underline{\Phi}\underline{x}(t_i) + \underline{B}_d\underline{u}(t_i) + \underline{E}_x\underline{n}_d(t_i) + \underline{w}_d(t_i) \tag{III-54a}$$

$$\underline{y}_c(t_i) = \underline{C}\underline{x}(t_i) + \underline{D}_y\underline{u}(t_i) + \underline{E}_y\underline{n}_d(t_i) \tag{III-54b}$$

In which $\underline{n}_d$ is a disturbance modeled by

$$\underline{n}_d(t_{i+1}) = \underline{\Phi}_n\underline{n}_d(t_i) + \underline{G}_{dn}\underline{w}_{dn}(t_i) \tag{III-55}$$

and $\underline{w}_{dn}$ a zero-mean white Gaussian noise of covariance $\underline{Q}_{dn}$ that is independent of $\underline{w}_d$. The controlled system modeled by Equation (III-54) is to duplicate, as closely as possible, the output of a command generator model

$$\underline{x}_m(t_{i+1}) = \underline{\Phi}_m\underline{x}_m(t_i) + \underline{B}_{dm}\underline{u}_m \tag{III-56a}$$

$$\underline{y}_m(t_i) = \underline{C}_m\underline{x}_m(t_i) + \underline{D}_m\underline{u}_m \tag{III-56b}$$

Mathematically, the objective of the CGT is to force the error

$$\begin{aligned}\underline{e}(t_i) &= \underline{y}_c(t_i) - \underline{y}_m(t_i)\\ &= [\underline{C}\ \underline{D}_y\ \underline{E}_y]\begin{bmatrix}\underline{x}(t_i)\\ \underline{u}(t_i)\\ \underline{n}_d(t_i)\end{bmatrix} - [\underline{C}_m\ \underline{D}_m]\begin{bmatrix}\underline{x}_m(t_i)\\ \underline{u}_m\end{bmatrix}\end{aligned} \tag{III-57}$$

to zero. When this command error is zero, the controlled system states and controls are tracking the ideal system trajectory (satisfying the original system dynamics less white noise, according to "Certainty Equivalence" ideas)

$$\underline{x}_I(t_{i+1}) = \underline{\Phi}\,\underline{x}_I(t_i) + \underline{B}_d\underline{u}_I(t_i) + \underline{E}_x\underline{n}_d(t_i) \qquad \text{(III-58)}$$

Another requirement is that the ideal system trajectory be a linear function of the model states, model controls, and disturbance states as shown by

$$\begin{bmatrix} \underline{x}_I(t_i) \\ \underline{u}_I(t_i) \end{bmatrix} = \begin{bmatrix} \underline{A}_{11} & \underline{A}_{12} & \underline{A}_{13} \\ \underline{A}_{21} & \underline{A}_{22} & \underline{A}_{23} \end{bmatrix} \begin{bmatrix} \underline{x}_m(t_i) \\ \underline{u}_m \\ \underline{n}_d(t_i) \end{bmatrix} \qquad \text{(III-59)}$$

where $\underline{u}_m(t_i) = \underline{u}_m$ for all t_i.

The solution of the CGT problem entails setting up and solving the matrix equations that the constants $\underline{A}_{11}$ through $\underline{A}_{23}$ in Equation (III-59) must satisfy. Using Equations (III-54), (III-58), (II-18) and (III-59) it has been shown that (Ref 18:153-155)

$$\begin{bmatrix} \underline{A}_{11} & \underline{A}_{12} & \underline{A}_{13} \\ \underline{A}_{21} & \underline{A}_{22} & \underline{A}_{23} \end{bmatrix} = \begin{bmatrix} \underline{\Pi}_{11} & \underline{\Pi}_{12} \\ \underline{\Pi}_{21} & \underline{\Pi}_{22} \end{bmatrix} \begin{bmatrix} \{\underline{A}_{11}(\underline{\Phi}_m - \underline{I})\} & \{\underline{A}_{11}\underline{B}_{dm}\} & \{\underline{A}_{13}(\underline{\Phi}_n - \underline{I}) - \underline{E}_x\} \\ \underline{C}_m & \underline{D}_m & -\underline{E}_y \end{bmatrix} \qquad \text{(III-60)}$$

Thus, in partitioned form, the equations to solve are

$$\underline{A}_{11} = \underline{\Pi}_{11}\underline{A}_{11}(\underline{\Phi}_m - \underline{I}) + \underline{\Pi}_{12}\underline{C}_m \qquad \text{(III-61a)}$$

$$\underline{A}_{12} = \underline{\Pi}_{11}\underline{A}_{11}\underline{B}_{dm} + \underline{\Pi}_{12}\underline{D}_m \qquad \text{(III-61b)}$$

$$\underline{A}_{13} = \underline{\Pi}_{11}\underline{A}_{13}(\underline{\Phi}_n - \underline{I}) - \underline{\Pi}_{11}\underline{E}_x - \underline{\Pi}_{12}\underline{E}_y \qquad \text{(III-61c)}$$

$$\underline{A}_{21} = \underline{\Pi}_{21}\underline{A}_{11}(\underline{\Phi}_m - \underline{I}) + \underline{\Pi}_{22}\underline{C}_m \qquad \text{(III-61d)}$$

$$\underline{A}_{22} = \underline{\Pi}_{21}\underline{A}_{11}\underline{B}_{dm} + \underline{\Pi}_{22}\underline{D}_m \qquad \text{(III-61e)}$$

$$\underline{A}_{23} = \underline{\Pi}_{21}\underline{A}_{13}(\underline{\Phi}_n - \underline{I}) - \underline{\Pi}_{21}\underline{E}_x - \underline{\Pi}_{22}\underline{E}_y \qquad \text{(III-61f)}$$

If $\underline{A}_{11}$ and $\underline{A}_{13}$ can be determined from Equations (III-61a) and (III-61c), which are of the form

$$\underline{X} = \underline{A}\,\underline{X}\,\underline{B} + \underline{C} \qquad \text{(III-62)}$$

(for which and algorithm for solution is available in Ref (1)), then the other terms are readily solved from these. Once the values of the $\underline{A}_{ij}$ are found, then the open loop command generator tracker control law is written from the lower partition of Equation (III-59) as

$$\underline{u}_I(t_i) = \underline{A}_{21}\underline{x}_m(t_i) + \underline{A}_{22}\underline{u}_m(t_i) + \underline{A}_{23}\underline{n}_d(t_i) \qquad \text{(III-63)}$$

$\underline{x}_I(t_i)$ can also be generated from Equation (III-59) as

$$\underline{x}_I(t_i) = \underline{A}_{11}\underline{x}_m(t_i) + \underline{A}_{12}\underline{u}_m(t_i) + \underline{A}_{13}\underline{n}_d(t_i) \qquad \text{(III-64)}$$

3.3.2 Closed Loop CGT/PI

It is useful to consider a PI closed loop law in conjunction with command generator tracking because (1) it forces the difference between actual system output and command generator model output to zero in steady state even in the face of some modeling errors, (2) it can accommodate unmodeled constant disturbances as well as reject the modeled ones, and (3) it will also turn out to include the model control's feedforward contribution to the actual system control signal. The four formulations for the closed-loop CGT/PI controller will now be presented. The first might be anticipated to be the most practical and give the best performance since it will assume the "pseudointegral" states are time varying, as they should be since the system states are time varying. The second assumes the "pseudointegral" states go from some initial equilibrium value to some new equilibrium value instantaneously with no time varying transient period. This would appear to give a faster transient response to the system. The third assumes that we have an open-loop CGT and a feedback PI controller and connect them in a feedback ad-hoc fashion. This method would seem to be logical but is not mathematically derivable (this method also adds a sixth gain to the system which will prevent it from being compatable with the program "PFEVAL" (Ref 21)). As a fourth formulation we can neglect this sixth gain, which is a direct feedthrough gain on the input $\underline{y}_m$. This would serve to slow down the response but would make the system compatable with the program "PFEVAL".

3.3.2.1 CGT/PI Formulation 1.

Assume that the open loop CGT law $\underline{u}_I(t_i)$ and the associated ideal state trajectory $\underline{x}_I(t_i)$ have been evaluated for all t_i as given by Equations (III-63) and (III-64). Now as done in Section 3.2.1, define perturbation variables as

$$\delta\underline{q}(t_i) = \underline{q}(t_i) - \underline{q}_I(t_i) \qquad \text{(III-65a)}$$

$$\delta\underline{x}(t_i) = \underline{x}(t_i) - \underline{x}_I(t_i) \qquad \text{(III-65b)}$$

$$\delta\underline{u}(t_i) = \underline{u}(t_i) - \underline{u}_I(t_i) \qquad \text{(III-65c)}$$

$$\delta\underline{y}_c(t_i) = \underline{y}_c(t_i) - \underline{y}_I(t_i) = \underline{y}_c(t_i) - \underline{y}_m(t_i) \qquad \text{(III-65d)}$$

and replacing $\underline{y}_d$ with $\underline{y}_m(t_i)$ in Equations (III-3), (III-4) and (III-5) we get

$$\begin{bmatrix} \delta\underline{x}(t_{i+1}) \\ \delta\underline{q}(t_{i+1}) \end{bmatrix} = \begin{bmatrix} \underline{\Phi} & \underline{0} \\ \underline{C} & \underline{I} \end{bmatrix} \begin{bmatrix} \delta\underline{x}(t_i) \\ \delta\underline{q}(t_i) \end{bmatrix} + \begin{bmatrix} \underline{B}_d \\ \underline{D}_y \end{bmatrix} \delta\underline{u}(t_i) \qquad \text{(III-66)}$$

Now, using the cost function of Equation (III-13), the optimal control solution using perturbation equations is as in Equation (III-14) and, returning to the original coordinate system the optimal control becomes

$$\begin{aligned} \underline{u}^*(t_i) = {} & \underline{u}_I(t_i) + \underline{G}_{c1}^*\underline{x}_I(t_i) + \underline{G}_{c2}^*\underline{q}_I(t_i) \\ & - \underline{G}_{c1}^*\underline{x}(t_i) - \underline{G}_{c2}^*\underline{q}(t_i) \end{aligned} \qquad \text{(III-67)}$$

As before, the issue is, what is the appropriate value of $\underline{q}_I(t_i)$. If we consider Equations (III-10a), (III-25) and (III-26a,b) we can let

$$\underline{q}_I(t_i) = - \underline{K}_{c22}{}^{-1}\underline{K}_{c12}{}^T\underline{x}_I(t_i) \qquad \text{(III-68)}$$

and, substituting this into Equation (III-67), we get the position form of the optimal controller

$$\begin{aligned} \underline{u}^*(t_i) = \underline{u}_I(t_i) &+ [\underline{G}_{c1}{}^* - \underline{G}_{c2}{}^*\underline{K}_{c22}{}^{-1}\underline{K}_{c12}{}^T]\underline{x}_I(t_i) \\ &- \underline{G}_{c1}{}^*\underline{x}(t_i) - \underline{G}_{c2}{}^*\underline{q}(t_i) \qquad \text{(III-69)} \end{aligned}$$

with

$$\underline{q}(t_{i+1}) = \underline{q}(t_i) + [\underline{y}_c(t_i) - \underline{y}_m(t_i)] \qquad \text{(III-70)}$$

To get the preferred incremental form of the complete CGT/PI controller we need to write Equation (III-69) at time t_i and t_{i-1}, explicitly subtract the two, substitute for $\underline{u}_I(t_i)$ and $\underline{x}_I(t_i)$ Equations (III-63) and (III-64) respectively, rearrange terms and utilize Equation (III-70) to get

$$\begin{aligned} \underline{u}^*(t_i) = \underline{u}^*(t_{i-1}) &- \underline{G}_{c1}{}^*[\underline{x}(t_i) - \underline{x}(t_{i-1})] \\ &+ \underline{G}_{c2}{}^*[\underline{y}_m(t_{i-1}) - \underline{y}_c(t_{i-1})] \\ &+ [\underline{A}_{21} + \underline{L}\underline{A}_{11}][\underline{x}_m(t_i) - \underline{x}_m(t_{i-1})] \\ &+ [\underline{A}_{22} + \underline{L}\underline{A}_{12}][\underline{u}_m(t_i) - \underline{u}_m(t_{i-1})] \\ &+ [\underline{A}_{23} + \underline{L}\underline{A}_{13}][\underline{n}_d(t_i) - \underline{n}_d(t_{i-1})] \qquad \text{(III-71)} \end{aligned}$$

where

$$\underline{L} = \underline{G}_{c1}^{*} - \underline{G}_{c2}^{*}\underline{K}_{c22}^{-1}\underline{K}_{c12}^{T} \qquad \text{(III-72)}$$

If we incorporate Equations (III-2) and (III-56b) into Equation (III-71) we get

$$\begin{aligned}\underline{u}^{*}(t_i) = {} & \underline{u}^{*}(t_{i-1}) - \underline{G}_{c1}^{*}[\underline{x}(t_i) - \underline{x}(t_{i-1})] \\ & + \underline{G}_{c2}^{*}\left\{[\underline{C}_m \ \underline{D}_m]\begin{bmatrix}\underline{x}_m(t_{i-1}) \\ \underline{u}_m(t_{i-1})\end{bmatrix} - [\underline{C} \ \underline{D}_y]\begin{bmatrix}\underline{x}(t_{i-1}) \\ \underline{u}(t_{i-1})\end{bmatrix}\right\} \\ & + [\underline{A}_{21} + \underline{L}\underline{A}_{11}][\underline{x}_m(t_i) - \underline{x}_m(t_{i-1})] \\ & + [\underline{A}_{22} + \underline{L}\underline{A}_{12}][\underline{u}_m(t_i) - \underline{u}_m(t_{i-1})] \\ & + [\underline{A}_{23} + \underline{L}\underline{A}_{13}][\underline{n}_d(t_i) - \underline{n}_d(t_{i-1})] \qquad \text{(III-73)}\end{aligned}$$

where $\underline{L}$ is as in Equation (III-72).

We have always assumed that

$$\underline{u}_m(t_i) = \begin{cases}\underline{u}_m = \text{constant} & i = 0,1,2,\ldots \\ \underline{u}_{m-} \neq \underline{u}_m & i = \ldots,-2,-1\end{cases}$$

i.e., that $\underline{u}_m$ is constant from t_o forward. For practical implementation, any time that $\underline{u}_m$ changes will cause i to be set to zero again. However, an inconsistency in the definition of the ideal trajectory, Equations (III-58) and (III-59), arises at the time of a step change in $\underline{u}_m$. It has been shown (Ref 18:161-162) that to avoid this inconsistency, whenever $\underline{u}_m(t_i)$ changes, we go back to time t_{i-1} and restart all variables under our control. This simply results in the command generator model, Equation (III-56a) becoming

$$\underline{x}_m(t_i) = \underline{\Phi}_m\underline{x}_m(t_{i-1}) + \underline{B}_{dm}\underline{u}_m(t_i) \qquad \text{(III-74)}$$

Incorporating this into Equation (III-73) will remove the inherent delay between system input and control output since the control is being applied one sample period earlier than before and yields the final CGT/PI control law as

$$\begin{aligned}\underline{u}^*(t_i) = \underline{u}^*(t_{i-1}) &- \underline{G}_{c1}^*[\underline{x}(t_i) - \underline{x}(t_{i-1})] \\ &+ \underline{G}_{c2}^*\left\{[\underline{C}_m\ \underline{D}_m]\begin{bmatrix}\underline{x}_m(t_{i-1})\\ \underline{u}_m(t_i)\end{bmatrix} - [\underline{C}\ \underline{D}_y]\begin{bmatrix}\underline{x}(t_{i-1})\\ \underline{u}^*(t_{i-1})\end{bmatrix}\right\} \\ &+ [\underline{A}_{21} + \underline{L}\underline{A}_{11}][\underline{x}_m(t_i) - \underline{x}_m(t_{i-1})] \\ &+ [\underline{A}_{22} + \underline{L}\underline{A}_{12}][\underline{u}_m(t_i) - \underline{u}_m(t_{i-1})] \\ &+ [\underline{A}_{23} + \underline{L}\underline{A}_{13}][\underline{n}_d(t_i) - \underline{n}_d(t_{i-1})] \qquad \text{(III-75)}\end{aligned}$$

where $\underline{L}$ is as given in Equation (III-72). (If we compare Equation (III-75) with Equation (II-18) we can see the similarities between the two CGT/PI control laws. These similarities are sufficient enough to allow the gains determined from this development of the CGT/PI control law based on the integral of the regulation error to fit directly into the CGT/PI/KF evaluation program "PFEVAL" without modifying "PFEVAL" in any way.)

3.3.2.2 CGT/PI Formulation 2.

Begin by considering the previous development up through Equation (III-67). As before, the issue is, what is the appropriate value of $\underline{q}_I(t_i)$. We could seek the best equilibrium $\underline{q}_{Io}$ and if we combine Equations (III-56a) and (III-56b) for the equilibrium condition $\underline{x}_m(t_{i+1})=\underline{x}_m(t_i)=\underline{x}_{mo}$ we get

$$\underline{y}_{mo} = \{\underline{C}_m(\underline{I}-\underline{\Phi}_m)^{-1}\underline{B}_{dm} + \underline{D}_m\}\underline{u}_m \tag{III-76}$$

If we assume this condition also applies for the transient period, we can combine Equation (III-68a) with Equation (III-26a), with $\underline{y}_d = \underline{y}_{mo}$ to get

$$\underline{q}_{Io} = \underline{q}_I(t_i) = -\underline{K}_{c22}{}^{-1}\underline{K}_{c12}{}^{T}\underline{\pi}_{12}\{\underline{C}_m(\underline{I}-\underline{\Phi}_m)^{-1}\underline{B}_{dm} + \underline{D}_m\}\underline{u}_m \tag{III-77}$$

Substituting this into Equation (III-67), we get the position form of the optimal controller as

$$\begin{aligned}\underline{u}^*(t_i) = \underline{u}_I(t_i) + \underline{G}_{c1}{}^*\underline{x}_I(t_i) - \underline{G}_{c2}{}^*\underline{K}_{c22}{}^{-1}\underline{K}_{c12}{}^{T}\underline{\pi}_{12} \\ \{\underline{C}_m(\underline{I}-\underline{\Phi}_m)^{-1}\underline{B}_{dm} + \underline{D}_m\}\underline{u}_m(t_i) - \underline{G}_{c1}{}^*\underline{x}(t_i) \\ - \underline{G}_{c2}{}^*\underline{q}(t_i)\end{aligned} \tag{III-78}$$

and Equation (III-70) still applies. To get the preferred incremental form of this CGT/PI controller we need to write Equation (III-78) at time t_i and t_{i-1}, explicitly subtract the two, substitute for $\underline{u}_I(t_i)$ and

$\underline{x}_I(t_i)$ Equations (III-63) and (III-64), respectively, rearrange terms and utilize Equations (III-70) and (III-74) to get

$$\begin{aligned}\underline{u}^*(t_i) = \underline{u}^*(t_{i-1}) &- \underline{G}_{c1}^*[\underline{x}(t_i) - \underline{x}(t_{i-1})] \\ &+ \underline{G}_{c2}^*\left\{[\underline{C}_m\ \underline{D}_m]\begin{bmatrix}\underline{x}_m(t_{i-1}) \\ \underline{u}_m(t_i)\end{bmatrix} - [\underline{C}\ \underline{D}_y]\begin{bmatrix}\underline{x}(t_{i-1}) \\ \underline{u}^*(t_{i-1})\end{bmatrix}\right\} \\ &+ [\underline{A}_{21} + \underline{G}_{c1}^*\underline{A}_{11}][\underline{x}_m(t_i) - \underline{x}_m(t_{i-1})] \\ &+ [\underline{A}_{22} + \underline{G}_{c1}^*\underline{A}_{12} - \underline{N}][\underline{u}_m(t_i) - \underline{u}_m(t_{i-1})] \\ &+ [\underline{A}_{23} + \underline{G}_{c1}^*\underline{A}_{13}][\underline{n}_d(t_i) - \underline{n}_d(t_{i-1})]\end{aligned}$$

(III-79)

where

$$\underline{N} = \underline{G}_{c1}^*\underline{K}_{c22}^{-1}\underline{K}_{c12}^T\underline{\Pi}_{12}\{\underline{C}_m(\underline{I}-\underline{\Phi}_m)^{-1}\underline{B}_{dm} + \underline{D}_m\} \quad \text{(III-80)}$$

3.3.2.3 CGT/PI Formulation 3.

Begin by assuming that we have the PI controller given by Equation (III-27) and the open-loop CGT given by Equation (III-63). If we put Equation (III-27) into terms of perturbation variables, replacing $\underline{y}_d$ with $\underline{y}_m$ we have

$$\underline{u}^*(t_i) = -\underline{G}_{c1}^*\delta\underline{x}(t_i) - \underline{G}_{c2}^*\delta\underline{q}(t_i) + \underline{E}\underline{y}_m(t_i) \quad \text{(III-81)}$$

and utilizing Equations (III-65a,b,c) we have

$$\begin{aligned}\underline{u}^*(t_i) - \underline{u}_I(t_i) = &-\underline{G}_{c1}^*[\underline{x}(t_i) - \underline{x}_I(t_i)] \\ &-\underline{G}_{c2}^*[\underline{q}(t_i) - \underline{q}_I(t_i)] \\ &+ \underline{E}\underline{y}_m(t_i)\end{aligned} \qquad \text{(III-82)}$$

Substituting in Equations (III-63) and (III-64), and rearranging terms we have the position form of this optimal controller as

$$\begin{aligned}\underline{u}^*(t_i) = &[\underline{A}_{21} + \underline{G}_{c1}^*\underline{A}_{11}]\underline{x}_m(t_i) \\ &+ [\underline{A}_{22} + \underline{G}_{c1}^*\underline{A}_{12}]\underline{u}_m(t_i) \\ &+ [\underline{A}_{23} + \underline{G}_{c1}^*\underline{A}_{13}]\underline{n}_d(t_i) \\ &- \underline{G}_{c1}^*\underline{x}(t_i) - \underline{G}_{c2}^*\underline{q}(t_i) + \underline{G}_{c2}^*\underline{q}_I(t_i) + \underline{E}\underline{y}_m(t_i)\end{aligned} \qquad \text{(III-83)}$$

To get the preferred incremental form of this CGT/PI controller we need to write Equation (III-83) at time t_i and t_{i-1}, explicitly subtract the two, utilize Equations (III-70) and (III-74), and let $\int \underline{q}_I = \underline{0}$, to get

$$\begin{aligned}\underline{u}^*(t_i) = &\underline{u}^*(t_{i-1}) - \underline{G}_{c1}^*[\underline{x}(t_i) - \underline{x}(t_{i-1})] \\ &+ \underline{G}_{c2}^*\left\{[\underline{C}_m \; \underline{D}_m]\begin{bmatrix}\underline{x}_m(t_{i-1}) \\ \underline{u}_m(t_i)\end{bmatrix} - [\underline{C} \; \underline{D}_y]\begin{bmatrix}\underline{x}(t_{i-1}) \\ \underline{u}^*(t_{i-1})\end{bmatrix}\right\} \\ &+ [\underline{A}_{21} + \underline{G}_{c1}^*\underline{A}_{11}][\underline{x}_m(t_i) - \underline{x}_m(t_{i-1})] \\ &+ [\underline{A}_{22} + \underline{G}_{c1}^*\underline{A}_{12}][\underline{u}_m(t_i) - \underline{u}_m(t_{i-1})] \\ &+ [\underline{A}_{23} + \underline{G}_{c1}^*\underline{A}_{13}][\underline{n}_d(t_i) - \underline{n}_d(t_{i-1})] \\ &+ \underline{E}\left\{[\underline{C}_m \; \underline{D}_m]\begin{bmatrix}\underline{x}_m(t_i) - \underline{x}_m(t_{i-1}) \\ \underline{u}_m(t_i) - \underline{u}_m(t_{i-1})\end{bmatrix}\right\}\end{aligned} \qquad \text{(III-84)}$$

3.3.2.4 CGT/PI Formulation 4.

Since the CGT/PI controller given by Equation (III-84) has six gains instead of the usual five of the other CGT/PI controllers, we could consider neglecting the sixth, that given by $\underline{E}$, which is a direct feedthrough speedup mechanism, and formulate a controller based on this simplification. (Note this has the same effect as neglecting the $\underline{N}$ term in Equation (III-79)). (This formulation will allow this use of the program "PFEVAL" (Ref 21), which requires only five gains as inputs.) The resulting controller is given by

$$\begin{aligned}\underline{u}^*(t_i) = \underline{u}^*(t_{i-1}) &- \underline{G}_{c1}^*[\underline{x}(t_i) - \underline{x}(t_{i-1})] \\ &+ \underline{G}_{c2}^*\left\{[\underline{C}_m\ \underline{D}_m]\begin{bmatrix}\underline{x}_m(t_{i-1})\\ \underline{u}_m(t_i)\end{bmatrix} - [\underline{C}\ \underline{D}_y]\begin{bmatrix}\underline{x}(t_{i-1})\\ \underline{u}^*(t_{i-1})\end{bmatrix}\right\} \\ &+ [\underline{A}_{21} + \underline{G}_{c1}^*\underline{A}_{11}][\underline{x}_m(t_i) - \underline{x}_m(t_{i-1})] \\ &+ [\underline{A}_{22} + \underline{G}_{c1}^*\underline{A}_{12}][\underline{u}_m(t_i) - \underline{u}_m(t_{i-1})] \\ &+ [\underline{A}_{23} + \underline{G}_{c1}^*\underline{A}_{13}][\underline{n}_d(t_i) - \underline{n}_d(t_{i-1})]\end{aligned} \quad \text{(III-85)}$$

3.4 Adding Implicit Model Following

In Chapter II, it was stated that implicit model following is achieved by including the model in a performance index that weights the error between system output derivatives and the desired model dynamics. A standard performance index is of the form of Equation (III-33)

$$J = \frac{1}{2}\int_{t_0}^{\infty} (\underline{x}^T \underline{\underline{X}}\underline{x} + \underline{u}^T \underline{\underline{U}}\underline{u})\, dt \qquad \text{(III-86)}$$

(There may or may not be a cross weighting term $\underline{\underline{S}}$)

where it is assumed that the system can be described as

$$\dot{\underline{x}} = \underline{\underline{A}}\underline{x} + \underline{\underline{B}}\underline{u} \qquad \text{(III-87)}$$

with output

$$\underline{y}_c = \underline{\underline{C}}\underline{x} \qquad \text{(III-88)}$$

Now, instead of the index of Equation (III-86), define a performance index which weights the error between system output derivatives and the model dynamics, where the model is

$$\dot{\underline{x}}_m = \underline{\underline{A}}_m \underline{x}_m \qquad \text{(III-89)}$$

and the corresponding performance index is

$$J_I = \frac{1}{2}\int_{t_0}^{\infty} [(\dot{\underline{y}}_c - \underline{\underline{A}}_m \underline{y}_c)^T X_I (\dot{\underline{y}}_c - \underline{\underline{A}}_m \underline{y}_c) + \underline{u}^T \underline{\underline{U}}_I \underline{u}]\, dt \qquad \text{(III-90)}$$

where the dimension of the output vector $\underline{y}_c$ and the model state vector $\underline{x}_m$ are equal. The weighting matrix $\underline{\underline{X}}_I$ weights errors between the output and model dynamics. Substituting Equations (III-87) and (III-88) into Equation (III-90) yields

$$J_I = \frac{1}{2}\int_{t_0}^{\infty} [(\underline{C}\underline{A}\underline{x} + \underline{C}\underline{B}\underline{u} - \underline{A}_m\underline{C}\underline{x})^T \underline{X}_I (\underline{C}\underline{A}\underline{x} + \underline{C}\underline{B}\underline{u} - \underline{A}_m\underline{C}\underline{x}) + \underline{u}^T\underline{U}_I\underline{u}]\, dt \qquad \text{(III-91)}$$

After collecting terms, Equation (III-91) leads to a performance index similar to Equation (III-86) but with a cross weighting term relating deviations in $\underline{x}$ and $\underline{u}$:

$$J_I' = \frac{1}{2}\int_{t_0}^{\infty} [\underline{x}^T\hat{\underline{X}}_I\underline{x} + 2\underline{x}^T\hat{\underline{S}}_I\underline{u} + \underline{u}^T\hat{\underline{U}}_I\underline{u}]\, dt \qquad \text{(III-92)}$$

where

$$\hat{\underline{X}}_I = (\underline{C}\underline{A} - \underline{A}_m\underline{C})^T\underline{X}_I(\underline{C}\underline{A} - \underline{A}_m\underline{C}) \qquad \text{(III-93a)}$$
$$\hat{\underline{S}}_I = (\underline{C}\underline{A} - \underline{A}_m\underline{C})^T\underline{X}_I\underline{C}\underline{B} \qquad \text{(III-93b)}$$
$$\hat{\underline{U}}_I = \underline{U}_I + \underline{B}^T\underline{C}^T\underline{X}_I\underline{C}\underline{B} \qquad \text{(III-93c)}$$

and $\underline{X}_I$ and $\underline{U}_I$ are symmetric with

$$\underline{X}_I \geq \underline{0} \qquad \text{(III-94a)}$$
$$\underline{U}_I > \underline{0} \qquad \text{(III-94b)}$$

If we rewrite Equation (III-92) as

$$J_I' = \frac{1}{2}\int_{t_0}^{\infty} \begin{bmatrix} \underline{x} \\ \underline{u} \end{bmatrix}^T \begin{bmatrix} \hat{\underline{X}}_I & \hat{\underline{S}}_I \\ \hat{\underline{S}}_I^T & \hat{\underline{U}}_I \end{bmatrix} \begin{bmatrix} \underline{x} \\ \underline{u} \end{bmatrix} dt \qquad \text{(III-95)}$$

and comparing this with Equation (III-33), letting the combined implicit/explicit cost function be

$$J_{IE} = J_I' + J_E \qquad \text{(III-96)}$$

(J_E = explicit cost function derived from the continuous cost function discussed previously), we have

$$J_{IE} = \frac{1}{2}\int_{t_0}^{t_{N+1}} \begin{bmatrix} \delta\underline{x} \\ \delta\underline{u} \end{bmatrix}^T \begin{bmatrix} \underline{X} + \hat{\underline{X}}_I & \underline{S} + \hat{\underline{S}}_I \\ (\underline{S} + \hat{\underline{S}}_I)^T & \underline{U} + \hat{\underline{U}}_I \end{bmatrix} \begin{bmatrix} \delta\underline{x} \\ \delta\underline{u} \end{bmatrix} dt \qquad \text{(III-97)}$$

Therefore, to add implicit model following, all we need do is add the appropriate implicit weights to the appropriate locations within the defining cost function. The overall cost weighting matrix thus becomes

$$J = \frac{1}{2}\int_{t_0}^{t_{N+1}} \begin{bmatrix} \delta\underline{x} \\ \delta\underline{q} \\ \delta\underline{u} \end{bmatrix}^T \begin{bmatrix} \underline{X} + \hat{\underline{X}}_I & \underline{0} & \underline{S} + \hat{\underline{S}}_I \\ \underline{0} & \underline{U}q & \underline{0} \\ (\underline{S} + \hat{\underline{S}}_I)^T & \underline{0} & \underline{U} + \hat{\underline{U}}_I \end{bmatrix} \begin{bmatrix} \delta\underline{x} \\ \delta\underline{q} \\ \delta\underline{u} \end{bmatrix} dt \qquad \text{(III-98)}$$

3.5 Summary

Chapter III has presented a theoretical development of the PI and CGT/PI controller with the PI controller based on the integral of the regulation error. The equations presented in this chapter were implemented in code and evaluated. The following chapters and appendices will look at the programs developed from these equations and compare them.

IV. CGT/PI/KF Design Computer Program

4.1 Introduction

The primary objective of this thesis effort is to create a computer program with which to design CGT/PI/KF controllers with the PI controller based on the integral of the regulation error. This chapter presents a general description of the program which has been developed --hereafter to be referred to as CGTPIQ--and incorporates a discussion of those portions of the program "CGTPIF" (Refs 9 and 21), upon which CGTPIQ is based, needed to add completeness of thought to the discussion. There were actually four programs developed to consider the four formulations of the CGT/PI presented in Chapter III. The discussion in this chapter pertains to the program developed which implements CGT/PI formulation 1 of Section 3.3.2.1. The other three programs are very similar as only a few lines of code needed to be changed to implement the other three CGT/PI formulations. (The required code changes are documented in Appendix C.)

While the specific test application for the program in the context of this thesis has been related to aircraft control design, CGTPIQ is written to be applicable to a wide variety of control design problems. It has the following attributes:

1. CGTPIQ executes interactively

2. The program utilizes efficient array allocation
    a. Initial memory allocation easily set
    b. Dynamic array allocation within total memory allocated
3. Various modes of entry are possible for the dynamics models
4. Design paths are automatically followed, with user prompts at necessary decision points
5. Requests for input include informative prompts
6. Copious program output is provided
    a. Output most relevant to design decisions are provided directly to the terminal
    b. Additional detailed output is provided to a separate output file
7. Information relevant to design iteration is preserved
8. Information needed for evaluation by computer program "PFEVAL" is preserved (Ref 21)
9. Error checking is performed, and messages given as appropriate

CGTPIQ employs computational routines available in a library of matrix computer routines described in Ref (13). Exclusive of the library routines, the program has a length of more than 2900 lines of source code. The programming language employed is ANSI Standard FORTRAN IV. Although the resulting source code is highly portable, local memory utilization limits for interactive execution may impose constraints. In use on a Control Data Corporation CYBER machine, the necessary load size was achievable with no impact on the source code. Thus the existing source code is in a pure form for whatever system. However, the final program size is much greater than the normal 65000 octal word limitation of the CYBER interactive system. Thus, in order to achieve interactive operation and provide sufficient free memory for array allocation so that problems of large and variable dimensions can be treated, a CYBER Loader option referred to as "Segmentation" (Ref 7) must be used. The required segmentation code will be presented in Appendix B. (Even with segmentation, the interactive program uses nearly 67000 octal words.)

CGTPIQ was written specifically to run interactively. Requests for input, while intentionally brief, tell the user what is expected of him--what, how many, and in what units, as appropriate. Output of information, relevant to design decisions, to the terminal is compact and automatically provided. Also, the user can determine the amount and category of output to the terminal in some cases, and according to need.

An objective of this thesis effort was for CGTPIQ to be similar to the program "CGTPIF" by Floyd, as modified by Moseley, and every effort was made for this to be the case. For this reason the two program's are almost identical in operation. The differences are presented in the appendices. Appendix A will give a description of only those routines that had to be changed in going from "CGTPIF" to "CGTPIQ. Appendix B will give a complete User's Guide to enable the user to use this program without having a copy of the thesis covering "CGTPIF".

4.2 Program Operating Principles and Organization

CGTPIQ has three design paths: (1) design of a PI controller (using explicit and/or implicit quadratic weights); (2) design of either an open-loop CGT or closed-loop CGT/PI controller, and (3) design of a Kalman filter. Corresponding to the first two design options is a controller evaluation set of routines, and corresponding to the third design option is a set of routines for filter evaluation. The evaluation routines perform the computations discussed in Appendix D.

A general flowchart of CGTPIQ is given in Figure IV-1, showing the main execution paths and design entries. The controller sample period is entered, the design model is established, and then the desired design path can be followed. The elementary design path choice is between controller and filter designs. The CGT or CGT/PI, and the PI controller (for explicit and/or implicit quadratic weights) design paths are then options within the controller design. If the PI controller design is pursued prior to the CGT design path, the CGT design will automatically be of the CGT/PI controller. If the PI controller gains have not been determined during the current execution of the program, then the designer may elect to design either a CGT or a CGT/PI controller. However, the CGT controller design is not

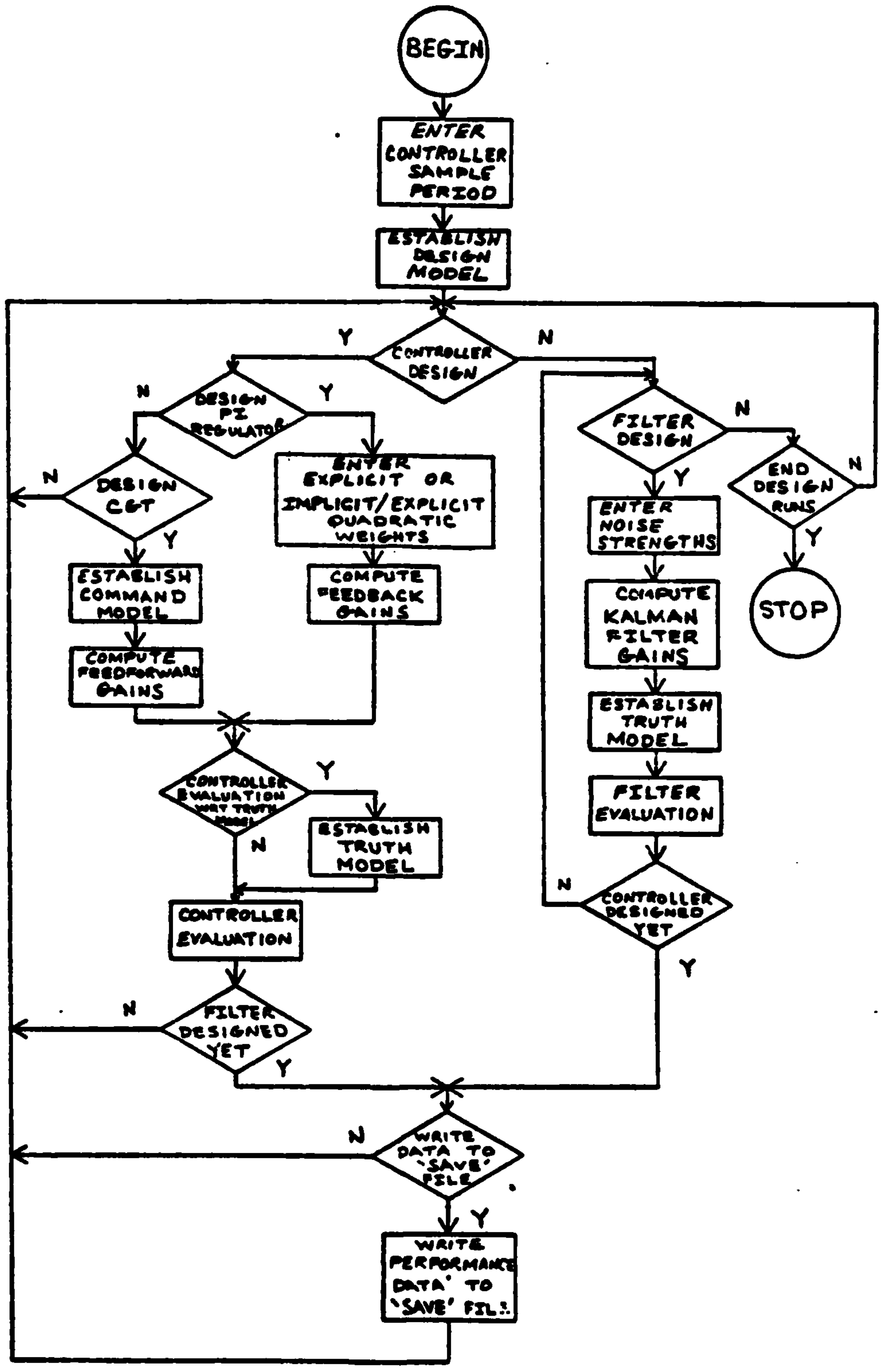


Fig. IV-1. CGTPIQ General Flowchart

pursued if the open-loop design model is unstable. The controller design path is followed automatically by the appropriate controller evaluation path. Similarly, the filter design path leads automatically to the filter evaluation. When the evaluation is complete the designer is given the opportunity to loop on the design path, choose a different design path, or terminate program execution.

4.3 Dynamics Models

The system for which a CGT/PI/KF controller is to be developed is assumed to be well represented by a set of linear, time invariant, stochastic state differential equations with zero-mean white Gaussian noise driving the system and/or disturbance states, and corrupting the system measurements. Such equations generally are derived as linearized perturbation equations for a non-linear system about a nominal operating point.

Three system models are employed: a "truth model", a "design model", and a "command model". The truth model is a model of the system which is as complete and accurate as possible for the control task under consideration. The design model is generally a simpler model and is the basis for the controller and filter gains and provides the set of states to which these gains are applied. While the quality of the resulting design is often developed by initially evaluating their performance with respect to the design model, their fine tuning to "true" performance potential in the real world environment must ultimately be with respect to the system truth model. The command model is a model representing the commands you wish the system to follow or the desirable characteristics you wish the system to emulate. Each of the three models is defined initially as a continuous-time system, then is discretized by the program for use within the program.

4.3.1 Design Model

The design model is given by

$$\dot{\underline{x}}(t) = \underline{A}\underline{x}(t) + \underline{B}\underline{u}(t) + \underline{E}_x\underline{n}_d(t) + \underline{G}\underline{w}(t) \qquad \text{(IV-1a)}$$

$$\dot{\underline{n}}_d(t) = \underline{A}_n\underline{n}_d(t) + \underline{G}_n\underline{w}_d(t) \qquad \text{(IV-1b)}$$

$$\underline{y}_c(t) = \underline{C}\underline{x}(t) + \underline{D}_y\underline{u}(t) + \underline{E}_y\underline{n}_d(t) \qquad \text{(IV-1c)}$$

$$\underline{z}(t_i) = \underline{H}\underline{x}(t_i) + \underline{H}_n\underline{n}_d(t_i) + \underline{v}(t_i) \qquad \text{(IV-1d)}$$

and $\underline{w}$, $\underline{w}_d$, and $\underline{v}$ are independent zero-mean noises with statistics

$$E\{\underline{w}(t)\underline{w}^T(t+\tau)\} = \underline{Q}\delta(\tau) \qquad \text{(IV-2a)}$$

$$E\{\underline{w}_d(t)\underline{w}_d^T(t+\tau)\} = \underline{Q}_n\delta(\tau) \qquad \text{(IV-2b)}$$

$$E\{\underline{v}(t_i)\underline{v}^T(t_j)\} = \underline{R}\delta_{ij} \qquad \text{(IV-2c)}$$

In these equations, $\underline{x}$, $\underline{u}$, $\underline{n}_d$, $\underline{y}_c$, and $\underline{z}$ are the system state, input, disturbance state, output, and measurement vectors, respectively. The dimensions of the model are

n = number of system states

r = number of system inputs

p = number of system outputs

m = number of system measurements

d = number of disturbance states

w = number of independent system noises

w_D = number of independent disturbance noises (IV-3)

and the dimensions of the matrices of the model are

$\underline{A}$:	n-by-n
$\underline{B}$:	n-by-r
$\underline{E}_x$:	n-by-d
$\underline{G}$:	n-by-w
$\underline{Q}$:	w-by-w
$\underline{C}$:	p-by-n
$\underline{D}_y$:	p-by-r
$\underline{E}_y$:	p-by-d
$\underline{H}$:	m-by-n
$\underline{H}_n$:	m-by-d
$\underline{R}$:	m-by-m
$\underline{A}_n$:	d-by-d
$\underline{G}_n$:	d-by-w_D
$\underline{Q}_n$:	w_D-by-w_D (IV-4)

A constraint to be imposed is that the number of design system inputs and outputs be equal: r=p. Also, the number of system states may not be less than the number of disturbance states, due to the computational setup used for the CGT solution. The dimensions n, r, and p must be non-zero; any of the other dimensions may be zero. If m is zero or if w and w_D are both zero, the Kalman filter design path cannot be pursued.

The design model is discretized at a specific fixed controller/filter sampling period T as follows. The disturbance state description is augmented to the system state model and discretized to form

$$\underline{x}_a(t_{i+1}) = \underline{\Phi}_a\underline{x}_a(t_i) + \underline{B}_{ad}\underline{u}(t_i) + \underline{w}_{ad}(t_i) \qquad \text{(IV-5)}$$

where "a" means augmented, and assuming the $\underline{u}$ is constant over a sample period,

$$\underline{\Phi}_a = e^{\underline{A}_a T} \qquad \text{(IV-6a)}$$

$$\underline{B}_{ad} = \int_0^T \underline{\Phi}_a(\tau)\underline{B}_a \, d\tau \qquad \text{(IV-6b)}$$

where $\underline{w}_{ad}$ is zero-mean white Gaussian discrete-time noise of discrete-time noise covariance

$$\underline{Q}_{ad} = \int_0^T \underline{\Phi}_a(\tau)\underline{G}_a\underline{Q}_a\underline{G}_a^T\underline{\Phi}_a^T(\tau)\,d\tau \qquad \text{(IV-6c)}$$

The matrices of Equations (IV-6a,b,c) may then be partitioned to the original component dimensions. Invoking the "Certainty Equivalence" property previously presented, the discrete-time models to be used for the deterministic controller design are the system state transition (propagation) equation, disturbance state equation, and output equation defined by

$$\underline{x}(t_{i+1}) = \underline{\Phi}\underline{x}(t_i) + \underline{B}_d\underline{u}(t_i) + \underline{E}_{xd}\underline{n}_d(t_i) \qquad \text{(IV-7a)}$$

$$\underline{n}_d(t_{i+1}) = \underline{\Phi}_n\underline{n}_d(t_i) \qquad \text{(IV-7b)}$$

$$\underline{y}_c(t_i) = \underline{C}\underline{x}(t_i) + \underline{D}_y\underline{u}(t_i) + \underline{E}_y\underline{n}_d(t_i) \qquad \text{(IV-7c)}$$

with $\underline{C}$, $\underline{D}_y$, and $\underline{E}_y$ as in Equation (IV-1c).

Henceforth, all equations relating to the controller design are considered deterministic.

4.3.2 Truth Model

The truth model is specified by

$$\dot{\underline{x}}_t(t) = \underline{A}_t\underline{x}_t(t) + \underline{B}_t\underline{u}_t(t) + \underline{G}_t\underline{w}_t(t) \qquad \text{(IV-8a)}$$

$$\underline{z}_t(t) = \underline{H}_t\underline{x}_t(t_i) + \underline{v}_t(t_i) \qquad \text{(IV-8b)}$$

$$\underline{x}'(t) = \underline{T}_{DT}\underline{x}_t(t) \qquad \text{(IV-8c)}$$

$$\underline{n}_d'(t) = \underline{T}_{NT}\underline{x}_t(t) \qquad \text{(IV-8d)}$$

and $\underline{w}_t$ and $\underline{v}_t$ are assumed to be independent zero-mean noises with

$$E\{\underline{w}_t(t)\underline{w}_t^T(t+\tau)\} = \underline{Q}_t\delta(\tau) \qquad \text{(IV-9a)}$$

$$E\{\underline{v}_t(t_i)\underline{v}_t^T(t_j)\} = \underline{R}_t\delta_{ij} \qquad \text{(IV-9b)}$$

where Equations (IV-8c) and (IV-8d) relate the system and disturbance states of the design model to the system states of the truth model. Note that $\underline{Q}_t$ and $\underline{R}_t$ are both assumed to be diagonal matrices. In these equations, $\underline{x}_t$, $\underline{u}_t$, and $\underline{z}_t$ are the truth model system state, input, and measurement vectors, respectively. The vectors $\underline{x}'$ and $\underline{n}_d'$ are as defined for the design model.

The dimensions of the truth model are

n_T = "number of system states"

r_T = "number of system inputs"

m_T = "number of system measurements"

w_T = "number of independent noises driving system dynamics" (IV-10)

and the dimensions of the matrices of the model are

$\underline{A}_t$: n_T-by-n_T

$\underline{B}_t$: n_T-by-r_T

$\underline{G}_t$: n_T-by-w_T

$\underline{Q}_t$: w_T-by-w_T

$\underline{H}_t$: m_T-by-n_T

$\underline{R}_t$: m_T-by-m_T

$\underline{T}_{DT}$: n-by-n_T

$\underline{T}_{NT}$: d-by-n_T (IV-11)

Dimensional compatibility for computations requires that the numbers of inputs and of measurements for the truth and design model be the same: $r_T=r$ and $m_T=m$. If the number of driving noises (w_T) is zero, evaluation of a Kalman filter design is not pursued (since a covariance analysis with no truth model driving noise would not be very informative).

The truth model is discretized for a specific controller/filter sampling period T, yielding

$$\underline{x}_t(t_{i+1}) = \underline{\Phi}_t\underline{x}_t(t_i) + \underline{B}_{td}\underline{u}_t(t_i) + \underline{w}_{td}(t_i) \qquad \text{(IV-12)}$$

where, for $\underline{u}$ constant over a sample period,

$$\underline{\Phi}_t = e^{\underline{A}_t T} \qquad \text{(IV-13a)}$$

$$\underline{B}_{td} = \int_0^T \underline{\Phi}_t(\tau)\underline{B}_t \, d\tau \qquad \text{(IV-13b)}$$

and $\underline{w}_{td}$ is zero-mean white Gaussian discrete-time noise with covariance

$$\underline{Q}_{td} = \int_0^T \underline{\Phi}_t(\tau)\underline{G}_t\underline{Q}_t\underline{G}_t^T\underline{\Phi}_t^T(\tau)\,d\tau \qquad \text{(IV-13c)}$$

4.3.3 Command Model

The command model is given by

$$\dot{\underline{x}}_m(t) = \underline{A}_m\underline{x}_m(t) + \underline{B}_m\underline{u}_m(t) \tag{IV-14a}$$

$$\underline{y}_m(t) = \underline{C}_m\underline{x}_m(t) + \underline{D}_m\underline{u}_m(t) \tag{IV-14b}$$

In these equations, $\underline{x}_m$, $\underline{u}_m$, and $\underline{y}_m$ are the command model state, input, and output vectors, respectively.

The dimensions of the command model are

n_M = "number of model states"

r_M = "number of model inputs"

p_M = "number of model outputs" (IV-15)

and the dimensions of the matrices are

$\underline{A}_m$: n_M-by-n_M

$\underline{B}_m$: n_M-by-r_M

$\underline{C}_m$: p_M-by-n_M

$\underline{D}_m$: p_M-by-r_M (IV-16)

Since it is desired to cause the system outputs to follow those of the command model, it is necessary that the number of outputs of the command and design models be equal: p_M=p. Also, the number of system states of the command model (n_M) cannot be greater that the number of system states of the design model (n). This constraint is due to the setup for computation of the CGT solution.

The discretized command model becomes

$$\underline{x}_m(t_{i+1}) = \underline{\Phi}_m\underline{x}_m(t_i) + \underline{B}_{md}\underline{u}_m(t_i) \tag{IV-17}$$

$$\underline{y}_m(t_i) = \underline{C}_m\underline{x}_m(t_i) + \underline{D}_m\underline{u}_m(t_i) \tag{IV-18}$$

where, for $\underline{u}_m$ constant over a sample period

$$\underline{\Phi}_m = e^{\underline{A}_m T} \tag{IV-19a}$$

$$\underline{B}_{md} = \int_0^T \underline{\Phi}_m(\tau)\underline{B}_m \, d\tau \tag{IV-19b}$$

and $\underline{C}_m$ and $\underline{D}_m$ are as before in Equation (IV-14b). (Note: the command model for explicit and implicit model following need not be the same, though they may be.)

4.3.4 Overall Model Dimensions

Problems of the following dimension can be accommodated by CGTPIQ

$$\begin{aligned}
n &\le 15\\
n+d &\le 15\\
r &\le 5\\
p &\le 5\\
m &\le 15\\
w &\le 15\\
w+w_D &\le 15\\
n_M &\le 10\\
r_M &\le 5\\
p_M &\le 5\\
n_t &\le 20\\
r_t &\le 5\\
m_t &\le 15\\
w_t &\le 20
\end{aligned} \tag{IV-20}$$

NOTE, other combinations of dimensions, some greater than and some less than these specific dimensions, will also be accommodated due to the method by which arrays are allocated and stored. If the user has a problem that does not fit these size restraints, he should try the problem to see if it will work. Messages will be printed if he has exceeded total allocation, and in that case, the basic program code will have to be altered to proceed.

4.4 Entry of Dynamics Models

Any of the three dynamics models described above may be established in any of three ways: (1) the dimensions and matrix elements may be entered directly from the terminal; (2) the dimensions and matrix elements may be entered from a "DATA" file; or (3) the dimensions and matrix elements may be established by user provided subroutines. When entered from the terminal directly or set by subroutines, only non-zero elements of the various matrices need be given. In many cases, this substantially simplifies establishing the matrices and reduces the probability of erroneous entries. Subsequent design runs for the same problem can then simply read the models from the "DATA" file previously created.

Before attempting to use the program the following should be noted:

1. The design model may be entered only once during a single program execution. Thus, in order to modify the design model, it is necessary to re-execute the program.

2. The truth and command models may be redefined as often as desired during a single program execution.

3. The SAVE and DATA files should be used to preserve the design, truth, and command models between distinct program executions. SAVE is a write only file and DATA is a read only file so information must be transferred from the SAVE to the DATA file between program executions (see Appendix B).

4. Models may be read from the DATA file repeatedly during program execution.

5. Changes in model dimensions require that all relevant defining matrices be reentered anew. Changes in dimension for a single model may require changes in dimensions for another due to the interrelationship of the different models. Also the matrices $\underline{T}_{DT}$ and $\underline{T}_{NT}$ may need to be changed.

4.5 Using the CGTPIQ Design Program

A few simple considerations of the operation of the design program particularly impact its ease of application to specific design problems. These and other more specific considerations are discussed fully in the "User's Guide" of Appendix B, and the reader is encouraged to refer to this manual before attempting to use the program. The following should be noted in addition to that discussed in Section 4.4 above:

1. Open-loop CGT designs can only be pursued in a given execution of the program if a PI controller design has not previously been accomplished and PI gains have not been read from the DATA file.

2. Any of the three design paths (PI controller, CGT controller, or Kalman filter) may be executed in any order and as often as desired during a single program execution. Results from each design path are preserved independently throughout execution.

3. In addition to the output available at the user's terminal, the same and much additional information is output to the file named 'LIST'. Following execution, this LIST file may be routed to a line printer for listing.

4. Plots of system time-response at the user terminal provide a very useful evaluation tool.

5. Regardless of the plot duration specified, the plot will include 50 time samples. These samples are uniformly distributed in time over the entire duration specified or automatically adjusted by the program.

6. In order to bypass the time-response evaluation (no plots to the terminal or to the LIST file) one need only specify that no plots are wished. For the CGT evaluation, specifying a zero index for the command input also bypasses the time-response evaluation.

7. The user must have a basic idea of how to design CGT/PI/KF controllers using this program before beginning execution so that questions asked of the user by the program can be answered correctly. This knowledge can be gained by reading the "User's Guide" of Appendix B.

4.6 Interpreting Plots to the Terminal

The plots of system response produced by CGTPIQ are of the "line printer" type. As printed at the terminal, the time axis runs vertically down the page with the initial time at the top-left margin and time points printed for each sample down the left margin. The dependent variables are plotted in the horizontal sense from left to right for increasing magnitude. Each variable plotted is marked by a distinct number (1 to 5) at each sample time. In the event that two variables occupy the same location in the plot field at the sample time, only the plot symbol of largest value will be marked at the point in question. Such coincidences of position can be inferred from the behavior of the variable with missing symbol at proximate time samples. Note that a special case of this is when two or more variables to be plotted actually represent the same variable.

The horizontal width of the plot field is 50 print positions. The scale of each plot is printed along the bottom margin and includes the values of the grid at each multiple of 10 print positions. In the usual case, each variable is plotted on its own unique scale and thus a scale range for each is given corresponding to the plot symbol used, but in some cases only a single scale is used for all variables plotted (i.e., anytime that the model output is requested to be plotted). Each plot will include an identifying title given it by the user.

4.7 Summary

This chapter has discussed the CGT/PI/KF design computer program by giving the models and some of the characteristics of the program. Chapter V will now demonstrate the use of the program to design CGT/PI/KF controllers. A complete sample run for a simple design problem is presented in the User's Guide, Appendix B. The main context of Chapter V will be a comparison of the four programs developed using the four formulations of the CGT/PI control law.

V. Analysis and Comparison of Design Results

5.1 Introduction

This chapter presents the results of the design and comparison efforts of this study. The design model upon which the analysis and comparisons were made is presented first. A baseline is then established by using the design model with the program "CGTPIF" (Refs 9 and 21), which designs a CGT/PI controller with the PI controller based on augmenting the original state equations with difference equations for control variables, treating control rates as driving functions, instead of augmenting the original state equations with the integrals of regulation error as done in "CGTPIQ". Only explicit model following is pursued in detail in this study since another thesis effort using "CGTPIF" with implicit model following is currently underway (Ref 20) and the results of that study are not available yet for comparison; however, it is anticipated that the formulation in this thesis may be very useful in conjunction with implicit model following, particularly since the proportional channel, $\underline{K}_p$, and the integral channel, $\underline{K}_i$, of the PI controller are not forced to be equal.
The same design model is used in all of these thesis efforts to provide continuity and comparison. (Note: it is recommended that this thesis, (Ref 20), should be required reading before using any of these software packages. There

were software errors in the original "CGTPIF" program (Refs 9 and 21) that were not discovered until "CGTPIQ" was programmed. These errors have now been corrected in the original "CGTPIF". Ref (20) documents these errors and gives a good insight into the effects of these errors on program execution. Also, the results of using implicit model following is thoroughly discussed in Ref (20)). This design model is then applied to the four programs developed in this study to determine the characteristics of each. A comparison is made between the results obtained from each of the four programs and a comparison with the baseline analysis. (The four programs are the programs which implement the four formulations for the closed-loop CGT/PI discussed in Chapter III, Section 3.3.2. All four of these programs employ the same PI controller.) The "best" of the four programs is then applied to a simple design problem in which constant disturbances, an alternate truth model which represents degraded performance due to misrepresentation of system dynamics in the design model, and initial conditions are applied to determine if this "best" design program can handle these situations. All designs are full-state feedback designs (no Kalman filter in the loop). The tables and time-response plots referred to in this chapter have been placed together at the end of the chapter to make the text easier to read.

5.2 Models

A CGT/PI controller for the Advanced Fighter

Technology Integration (AFTI) F-16 was chosen as the example controller to be designed using each of the programs. The AFTI F-16 is an aircraft which has been modified for advanced flight control research. Through the use of two independently controlled longitudinal flight control surfaces (a horizontal tail and a trailing edge wing flap), it is possible to achieve direct control of the aircraft's pitch attitude without changing its flight path angle and, hence, its flight trajectory. This capability is expected to be very useful in situations such as air-to-air gunnery, in which the attacker must match the target's trajectory while simultaneously achieving the required gun lead angle in the pitch plane. Such a design was of interest for this study, since it inherently requires the use of multiple-input multiple-output (MIMO) design methodology to achieve decoupled control of the two outputs and maintain closed-loop stability with a plant (the F-16 aircraft) which is unstable.

The efforts of this study and comparison used a five-state design model for the AFTI F-16 (Refs 8 and 9). The model represents the linearized flight characteristics of the aircraft operating at 0.8 mach at 10,000 feet. This particular flight condition is representative of an operating point within the usual air-to-air combat flight regime. Since the AFTI F-16 is unstable in pitch in this regime, open-loop CGT designs are infeasible. Therefore, only CGT/PI designs were pursued. The design model is a time-invariant, five state model of the form

$$\dot{\underline{x}}(t) = \underline{A}\underline{x}(t) + \underline{B}\underline{u}(t) \tag{V-1}$$

where the state vector $\underline{x}$ consists of

$x(1)$ = pitch angle (degrees)

$x(2)$ = angle of attack (degrees)

$x(3)$ = pitch rate (degrees per second)

$x(4)$ = horizontal tail deflection (degrees)

$x(5)$ = trailing edge flap deflection (degrees)

$x(1) - x(2)$ = flight path angle (degrees) (V-2)

and

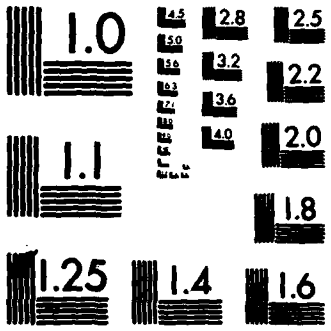
$$\underline{A} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ -1.08E-3 & -1.7 & 0.994 & -0.179 & -0.295 \\ 0 & 5.93 & -0.668 & -25.3 & -5.88 \\ 0 & 0 & 0 & -20.0 & 0 \\ 0 & 0 & 0 & 0 & -20.0 \end{bmatrix} \tag{V-3}$$

$$\underline{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 20.0 & 0 \\ 0 & 20.0 \end{bmatrix} \tag{V-4}$$

(Note: In this design model, actuators are represented as having characteristics of a first-order lag response to commands.) The outputs are expressed as

AD-A138 453 COMMAND GENERATOR TRACKER SYNTHESIS METHODS USING AN LQG (LINEAR SYSTEM M..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. J P MCMILLIAN 2/4

UNCLASSIFIED DEC 83 AFIT/GE/EE/83D-46 F/G 9/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

$$\underline{y}_c(t) = \underline{C}\underline{x}(t) \tag{V-5}$$

where

$$\underline{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 \end{bmatrix} \tag{V-6}$$

indicating that the outputs of interest are the pitch angle (y1=x1) and the flight path angle (y2=x1-x2). The limits to be considered in the design included a maximum deflection of 25 degrees in either direction at a maximum rate of 60 degrees per second for the horizontal tail, and +20 degrees to -23 degrees at a maximum rate of 52 degrees per second for the flap (Ref 9).

A two state command model was used in the design with

$$\underline{A}_m = \begin{bmatrix} -5 & 0 \\ 0 & -5 \end{bmatrix} \tag{V-7}$$

$$\underline{B}_m = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix} \tag{V-8}$$

$$\underline{C}_m = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{V-9}$$

and $\underline{D}_m = \underline{0}$. The command model was chosen to represent the ideal output characteristics of decoupled first-order responses for the pitch and flight path angles.

5.3 Design Analysis

5.3.1 Introduction

The design analysis consists of sets of time response plots, tables of PI and CGT/PI gains, and PI pole locations for several different sets of quadratic weights applied to each of the five programs (CGTPIF and the four versions of CGTPIQ).

For CGTPIF, initial quadratic weights of one each were entered for output deviations (Y), control magnitude (UM), control rates (UR), and the (3,3) element of the resultant state weighting $\underline{X}$ matrix (to limit the pitch rate magnitude). Each of these was increased one at a time to ten and another set of time response plots made. By noting the effects of changing the weightings, one at a time, the designer will get a feel for what weights to apply to get the effects he desires. A final run is made with CGTPIF using a "best" selected set of weights. Each of these runs is discussed in the following subsections. (A set of time response plots using CGTPIF consists of first the PI time response and second the corresponding CGT/PI time response using the just-determined PI gains.)

For CGTPIQ, initial quadratic weights of one were entered for output deviations (Y), integral of the regulation error (UQ), control magnitudes (UM), and the (3,3) element of the resultant state weighting $\underline{X}$ matrix. (Note: the selection of one for each weight has no correlation with the designs using CGTPIF other than to give a starting point for the discussion.) Each of these was increased one at a time to ten and another set of time response plots made. Again, by noting the effects of changing the weightings, one at a time, the designer will get a feel for what weights to apply to get the effects he desires. Three additional runs were made using CGTPIQ. The first was obtained by using the same set of weights as the "best" run made using CGTPIF with the weights for UR in CGTPIF applied to UQ in CGTPIQ, all other weights remaining the same. This was done to see if in fact there was any direct correlation between the two designs. The second was achieved by applying weights to every diagonal element in the $\underline{X}$ matrix (i.e., adding quadratic cost associated with the two actuator states. A weight of one was arbitrarily selected for each of these.) and the third by applying explicit and implicit weights just to see the effect. (A set of time responses using CGTPIQ consists first of the PI time response followed by four CGT/PI time responses using each of the four formulations presented in Chapter III, each based on the same set of PI gains just determined.)

All PI time response plots represent the same four variables. Symbol '1' represents the pitch angle, symbol '2' the flight path angle, '3' is the position of the horizontal tail and '4' is the position of the trailing edge flaps. Each PI time response represents the response to an initial condition of one degree in the pitch angle state.

All CGT/PI time response plots represent the same three variables. Symbol '1' represents the pitch angle, symbol '2' is the model output that the pitch angle is to follow, and '3' is the flight path angle which is desired to stay at zero. Each CGT/PI time response represents the response to a step input of magnitude one for the model. All state initial conditions are zero for the CGT/PI.

5.3.2 CGTPIF Baseline Design

The baseline design using CGTPIF is shown in Figures V-1 through V-12 and in Table V-1. Figure V-1 represents the PI controller for quadratic weight set (abbreviated QWS in all of the figures) number one of Table V-1. Figure V-2 is the corresponding CGT/PI for the same quadratic weight set. Figures V-3 and V-4 are the PI and CGT/PI, respectively, corresponding to quadratic weight set number two of Table V-1. This two plot set for one quadratic weight set correspondence continues up to Figures V-11 and V-12 corresponding to quadratic weight set six of Table V-1.

Observation of the CGT/PI performance for all six sets of quadratic weights indicates a good model following of the pitch angle (there are slight differences in the transient performance of each) with very little deviation in the flight path angle. Therefore, the only observations that will be discussed in detail is how the quadratic weights affected the motion of the horizontal tail and trailing edge flap to achieve this good model following of the CGT/PI. To determine this we need to look at the PI time response for each set of quadratic weights. Figure V-1 will be used as a base to judge the effects of increasing each quadratic weight.

Increasing the weight on the output deviation (Y) had very slight effect on the performance, as shown in Figure V-3, but did show a slight decrease in rise time with a corresponding larger overshoot as would be expected. (Note: due to the effects of scaling of each variable on a time response plot, the designer will need to be careful, in that, what looks to be a "better" or "worse" response may just be due to a different scale for that variable from one run to the next. This is not that critical for this first comparison but is critical on some of the other CGT/PI time response plots.)

Increasing the weight on the (3,3) element of the $\underline{X}$ weighting matrix (i.e., on pitch rate) again had very little effect, as shown in Figure V-5, but did decrease the rise time and settling time, as expected, with a corresponding greater overshoot.

Increasing the weight on the control magnitude (UM) had the effect of slowing down the horizontal tail motion but greatly increasing the speed and deflection of the trailing edge flap as shown in Figure V-7. This difference in effect on the two control surfaces was not expected since weighting the control for each should have allowed each to move faster and further initially.

Increasing the weight on the control rate (UR) had the effect of slowing down both the tail and flap motion as expected, producing very little overshoot with trailing edge flap motion almost within tolerance limits as shown in

Figure V-9.

As a last set of quadratic weights to be used which were considered the "best" for this application, the weights (Y) and X(3,3) were increased to decrease the rise time, thereby speeding up the motion of the flaps and tail, but at the same time the weight (UR) was increased to limit the deflection and overshoot of each. This combination produced flap and tail motion that stayed in limits and gave good model following characteristics to the CGT/PI while maintaining the flight path angle near zero, as shown in Figures V-11 and V-12.

The corresponding PI pole motion for each of the sets of quadratic weights is shown in Table V-1 along with the corresponding PI and CGT/PI gains needed to produce the responses noted. The PI pole motion may give insight into design characteristics but this is not discussed here. The pole locations are included in the table since some readers/designers may wish to know this information.

5.3.3 <u>CGTPIQ Design Analysis</u>

The design and comparisons using the four versions of CGTPIQ are presented in Figures V-13 through V-52 and Table V-2. There are eight sets of five time response plots. The first plot of each set is the PI time response for the set of quadratic weights shown in Table V-2. Plot two of each set is the CGT/PI time response produced using CGT/PI formulation 1. Similarly, the third, fourth and fifth plots of each set are the CGT/PI time responses produced using CGT/PI formulations 2, 3, and 4, respectively. Each CGT/PI is based on the same PI gains previously determined for that set.

The determination of a "best" CGT/PI out of the four possible will be the one that most closely approaches the performance noted by using CGTPIF. This was very easy to determine, for, if we notice, every CGT/PI based on formulations 1 and 2, no matter what the quadratic weights were that were applied to the PI, showed an inherent lag between model output and system output. This lag was greater for the CGT/PI based on formulation 2. (If we recall from Chapter III, this was not supposed to be the case when comparing these two. The cause of this characteristic is not yet resolved.) The CGT/PI based on CGT/PI formulation 3 showed an inherent overshoot due to faster transient performance for all quadratic weight sets. This was to be expected, as noted in Chapter III, due to the extra gain, $\underline{E}$, providing a direct feedthrough of the input. Only the ad-hoc formulation 4 (based on neglecting the terms, $\underline{E}\delta\underline{y}_m$, from formulation 3 which produced a fast transient and also the term, $\underline{N}$, from formulation 2, which turns out to be the term which slows down the transient response) yields a CGT/PI controller that appears to give satisfactory performance (satisfactory in the sense that it most closely approaches the performance noted using CGTPIF) for a given PI set of quadratic weights. What we need to look at now is what effect the different sets of quadratic weights, as presented in Table V-2, have on the PI performance.

We will use Figure V-13, which is the PI time response based on quadratic weight set 1 of Table V-2, as the base from which to determine the effects of increasing each quadratic weight. The first thing that is noted in looking at Figure V-13 is the much slower response of this PI controller compared to the one used in CGTPIF. This slowness shows in every PI time response for every set of quadratic weights. (This may or may not be a good trait depending on the desires of the designer.) But, at the same time, it is noted that the motion of the flaps and tail are not nearly as harsh as in the PI controller produced by CGTPIF. This would be important for saturation concerns and for robustness (these issues are amoung those being investigated by Capt Miller, Ref (20), using the program "CGTPIF").

Increasing the weight on the output deviation (Y) had very little effect on the system as shown in Figure V-18. This same response was noted using CGTPIF as mentioned in the previous section.

Increasing the weight on the (3,3) element of the $\underline{X}$ matrix had a very definite slowing effect on the system and the coresponding reduction in overshoot as shown in Figure V-23. This was as expected due to the increased weight on the pitch rate. Also, the motion of the horizontal tail is not nearly as smooth.

Increasing the weight on the integral of the regulation error (UQ) had the effect of speeding up the system with the corresponding overshoot produced, as shown in Figure V-28. (All things considered, this is not a bad PI design in itself and it produces acceptable CGT/PI performance as shown in Figure V-32.)

Increasing the weight on the control magnitude (UM) had the effect of greatly slowing down the system, as expected, with the motion of the flap and tail greatly reduced, as shown in Figure V-33.

Figure V-38 shows the PI time response for the same set of quadratic weights as the "best" design using CGTPIF. In this case this is not a bad design either as response time is reasonable and flap and tail motion are maintained within the required limits. The CGT/PI also shows satisfactory results with good model following and flight path angle deviations very slight, as shown in Figure V-42. The purpose of this test was to determine if there was a weight that could be applied to UQ, all other weights remaining the same as in CGTPIF, that would produce a response exactly like that obtained using CGTPIF. Although the response was similar, we were not able to find a weight for UQ that would produce the exact results.

The last two sets of quadratic weights were used to determine if perhaps additional weights could be added, either manually in the $\underline{X}$ matrix or by using implicit weights, which also fills in the $\underline{X}$ matrix with off-diagonal elements, to improve the performance of the PI controller (making it more closely approximate the good performance as seen using CGTPIF). Nothing spectacular was noted in these two as shown in Figures V-43 and V-48 but it was noted that adding small implicit weights did give performance similar to that obtained by manually adding additional weights in the $\underline{X}$ matrix, thereby relieving the designer of the burden of guessing weights to add. This is an expected characteristic of using implicit model following.

The corresponding PI pole motion for each of the sets of quadratic weights is shown in Table V-2 along with the corresponding PI and CGT/PI gains needed to produce the responses noted. The PI proportional gains, $\underline{K}_p$, and integral gains, $\underline{K}_i$, are also listed. As expected, they are different, and therefore are a potentially very useful tool in design work. By being able to adjust the proportional and integral gains, the designer would be able to compensate for a bad trait of PI controllers known as wind-up. This was not investigated in this study and is left for follow-on work.

5.4 Simple Design Problem Using CGT/PI Formulation 4

Since CGT/PI formulation 4 was shown to be a candidate for a good CGT/PI controller, it was decided to test this controller's performance against adverse conditions. A simple design problem was selected that contained disturbance states, an alternate truth model used to evaluate the degradation in perfromance due to misrepresented system dynamics in the design model, and initial condi·ions. :his is the same simple design model as presented in Fl·yd's thesis (Ref 9), and is presented here as

Design Model

$$\dot{\underline{x}}(t) = \begin{bmatrix} -2 & 5 \\ -5 & -2 \end{bmatrix} \underline{x}(t) + \begin{bmatrix} 3 & -1 \\ 1 & 5 \end{bmatrix} \underline{n}_d(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \underline{u}(t) \qquad \text{(V-10)}$$

$$\dot{\underline{n}}_d(t) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \underline{n}_d(t) \qquad \text{(V-11)}$$

$$\underline{y}_c(t) = [1 \quad 0] \underline{x}(t) \qquad \text{(V-12)}$$

$$\underline{n}_d(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} -1 \\ 2 \end{bmatrix} \qquad \text{(V-13)}$$

Truth Model

$$\dot{\underline{x}}_t(t) = \begin{bmatrix} -2 & 5 & 3 & -1 \\ -5 & -2 & 1 & 5 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \underline{x}_t(t) + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \underline{u}_t(t) \qquad \text{(V-14)}$$

Alternate Truth Model

$$\dot{\underline{x}}'_t(t) = \begin{bmatrix} -1 & 5 & 3 & -1 \\ -5 & -1 & 1 & 5 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \underline{x}'_t(t) + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \underline{u}_t(t) \qquad \text{(V-15)}$$

Command Models

First order command model (CM01):

$$\dot{\underline{x}}_m(t) = -5\underline{x}_m(t) + \underline{u}_m(t) \qquad \text{(V-16)}$$

$$\underline{y}_m(t) = 5\underline{x}_m(t) \qquad \text{(V-17)}$$

Second order command model (CM02):

$$\dot{\underline{x}}_m(t) = \begin{bmatrix} -5 & 5 \\ -5 & -5 \end{bmatrix} \underline{x}_m(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \underline{u}_m(t) \qquad \text{(V-18)}$$

$$\underline{y}_m(t) = [10 \quad 0] \underline{x}_m(t) \qquad \text{(V-19)}$$

Figure V-53 shows an open-loop CGT design (1 is the system output and 2 is the command model output) to demonstrate the effects of adding the PI feedback controller design of Figure V-54 (1 is state x2, 2 is the system output, and 3 is the applied control). Figure V-55 shows the improved performance of the closed-loop CGT/PI over the open-loop CGT. This figure is based on the first order command model (CM01). When the same PI controller is applied, and the second order command model (CM02) selected, the performance is drastically improved as shown in Figure V-56. (There is no difference between model and system outputs.) These last four time response plots are based on the nominal truth model. Figure V-57 shows the effect on the CGT/PI of adding a constant disturbance and Figure V-58 shows the effect of not only having disturbances but also the alternate truth model. The performance of these two CGT/PI controllers compares favorable with that obtained by Floyd when using the program "CGTPIF" (Ref 9). As a final test, Figure V-59 adds initial conditions to the design with disturbance and alternate truth model. Note the CGT/PI controller takes a little time to recover from all the discrepancies, but it does recover and gives good performance as steady-state conditions are approached.

5.5 Summary

Chapter V has presented and compared each of the PI and CGT/PI controllers discussed in this thesis through use of the programs which were developed to implement them. A comparison was also made with the original program "CGTPIF".

Initial indications are that the program "CGTPIF" has certain advantages over "CGTPIQ" in that it offers greater control over the PI controller, but this does not necessarily mean better performance in the CGT/PI controller when we note the performance due to CGT/PI formulation 4. This particular CGT/PI controller was shown to be able to generate designs with desirable closed-loop characteristics and to handle adverse conditions just as well as the CGT/PI controller implemented in the program "CGTPIF".

Table V-1: CGTPIF Gains and Pole Positions for Various PI Quadratic Weights

Wts Y;X33;UM;UR	KX	KZ	PI Poles	KXM	KXU
1,1;1;1,1;1,1 1	-62.21 30.86 -3.13 2.19 0.12 124.5 -134.1 0.57 0.12 1.69	0.39 1.17 -4.7 4.64	-1.51+j0.45 -18.1+j13.6 -56,-20,-55	-14.13 -24.43 -27.20 103.4	-12.22 -11.43 -0.71 48.96
10,10;1;1,1;1,1 2	-70.33 32.42 -3.34 2.25 0.13 124.8 -135.1 0.55 0.13 1.70	-0.9 -1.6 -4.9 4.93	-18+j13.62 -2.76+j0.79 -20,-55,-56	-19.62 -25.94 -27.82 104.4	-13.24 -11.52 -0.80 49.02
1,1;10;1,1;1,1 3	-104.2 30.55 -4.77 2.66 0.23 114.8 -134.2 0.19 0.23 1.72	0.47 -1.1 -4.7 4.66	-24.8+j28.9 -0.45,-1.74, -62,-55	-48.48 -24.24 -35.17 103.5	-20.04 -11.41 -2.53 48.96
1,1;1;10,10;1,1 4	-79.24 56.58 -3.85 3.21 0.04 236.4 -246.2 1.48 0.03 3.07	1.54 -2.0 -8.6 8.48	-1.44+j46.1 -74.5+j157 -10,-17,-20	-0.45 -46.22 -40.5 197.6	-14.73 -17.83 2.78 76.5
1,1;1;1,1;10,10 5	-26.24 12.52 -1.65 1.29 0.14 50.87 -54.99 0.15 0.14 0.72	0.15 -0.5 -1.9 1.90	-12.7+j12.7 -1.5+j45.2 -26,-16,-20	-4.74 -8.82 -13.56 37.0	-5.76 -6.91 -2.11 29.54
5,5;2;1,1;10,10 6	-32.34 12.92 -1.96 1.44 0.17 50.08 -55.29 0.09 0.17 0.72	-0.15 -0.6 -1.99 2.0	-13.56+j15.5 -1.87+j0.7 -28,-16,-20	-8.93 -9.23 -14.4 37.3	-7.21 -6.97 -2.39 29.57

```
  0.00     +   3     +         +         +     2   1     4   +
   .02     +4        +         +         +  2     1+    3    +
   .04     4         +         +      2  +    1  3 +         +
   .06     +        4+         + 2 3    1+         +         +
   .08     +         + 3    2 4+  1      +         +         +
   .10     +    3    +  2   1  +      4  +         +         +
   .12     + 3       2 1       +         + 4       +         +
   .14     +3      2 +         +         +      4  +         +
   .16     +3   12   +         +         +         4         +
   .18     +3 1 2 : :+: : : : :+: : : : :+: : : : :+:4: : : :+
   .20     + 3 2     +         +         +         +  4      +
   .22     + 32      +         +         +         +   4     +
   .24     + 13      +         +         +         +    4    +
   .26     + 23      +         +         +         +    4    +
   .28     + 23      +         +         +         +     4   +
   .30     + 2 3     +         +         +         +     4   +
   .32     + 2 3     +         +         +         +     4   +
   .34     + 213     +         +         +         +     4   +
   .36     + 213     +         +         +         +     4   +
   .38     +:213: : :+: : : : :+: : : : :+: : : : :+: : :4: :+
   .40     +2 13     +         +         +         +     4   +
   .42     +2 13     +         +         +         +     4   +
   .44     +2 13     +         +         +         +     4   +
   .46     +2 13     +         +         +         +     4   +
   .48     +2 13     +         +         +         +     4   +
   .50     +2 3      +         +         +         +     4   +
   .52     +2 3      +         +         +         +     4   +
   .54     +2 31     +         +         +         +     4   +
   .56     +2 31     +         +         +         +     4   +
   .58     +2 31: : :+: : : : :+: : : : :+: : : : :+: : :4: :+
   .60     +2 31     +         +         +         +     4   +
   .62     +2 31     +         +         +         +     4   +
   .64     +2 31     +         +         +         +     4   +
   .66     +2 31     +         +         +         +     4   +
   .68     +2 31     +         +         +         +     4   +
   .70     +2 31     +         +         +         +     4   +
   .72     +2 31     +         +         +         +     4   +
   .74     +2 31     +         +         +         +     4   +
   .76     +2 31     +         +         +         +     4   +
   .78     +2 31: : :+: : : : :+: : : : :+: : : : :+: : :4: :+
   .80     +2  3     +         +         +         +     4   +
   .82     +2  3     +         +         +         +     4   +
   .84     +2  3     +         +         +         +     4   +
   .86     +2  3     +         +         +         +     4   +
   .88     +2  31    +         +         +         +     4   +
   .90     +2  31    +         +         +         +     4   +
   .92     +2  31    +         +         +         +     4   +
   .94     +2  31    +         +         +         +     4   +
   .96     +2  31    +         +         +         +     4   +
   .98     +2 :31 : :+: : : : :+: : : : :+: : : : :+: : :4: :+
SCALE 1   -.2000     .1000     .4000     .7000    1.0000    1.3000
SCALE 2   -.1000     .2000     .5000     .8000    1.1000    1.4000
SCALE 3  -2.0000    3.0000    8.0000   13.0000   18.0000   23.0000
SCALE 4 -42.0000  -33.0000  -24.0000  -15.0000   -6.0000    3.0000
```

Figure V-1. PI for QWS-1 (CGTPIF)

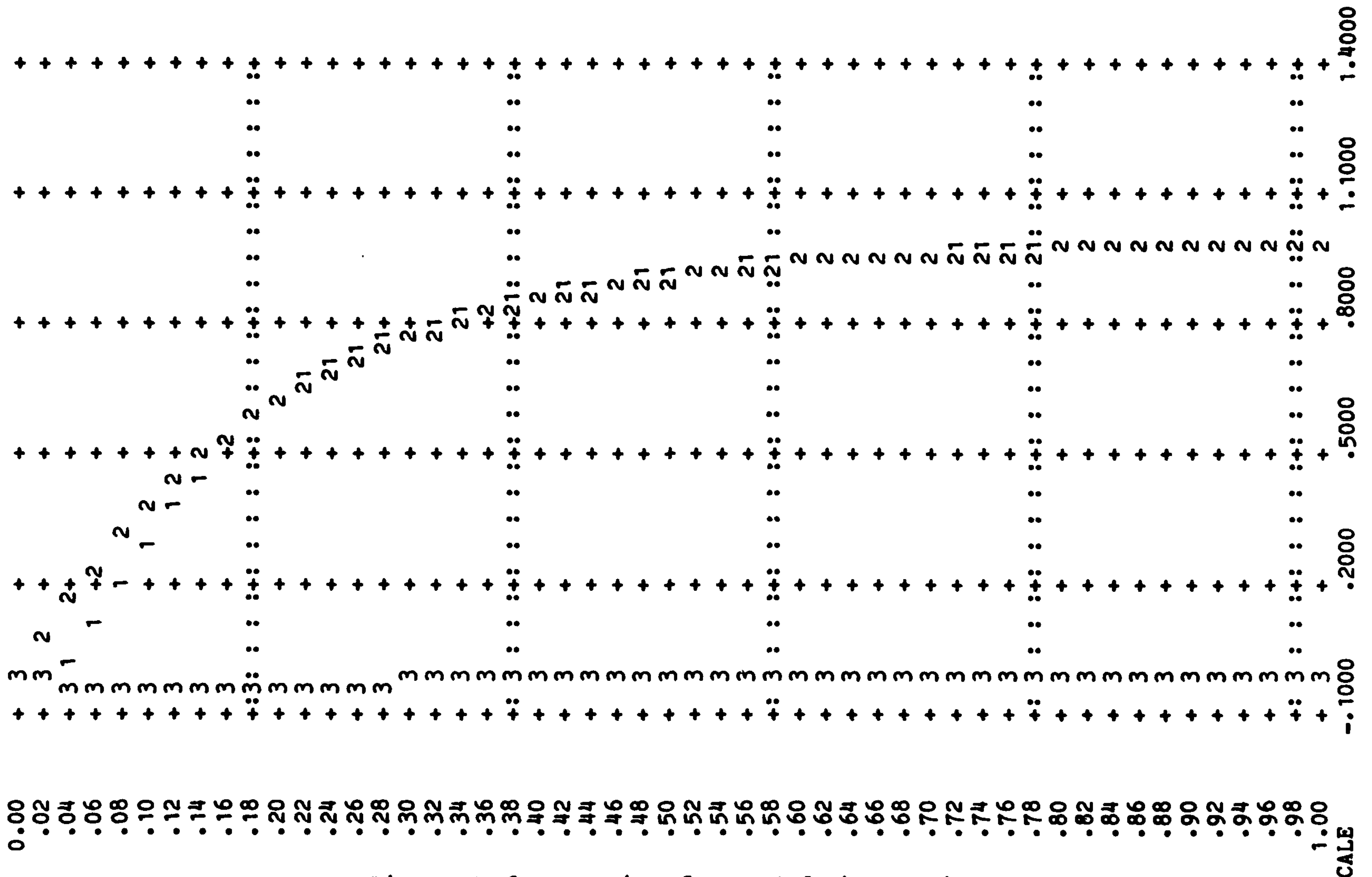


Figure V-2. CGT/PI for QWS-1 (CGTPIF)

```
   0.00      +    3    +         +         +     2   +  1   4  +
    .02      + 4       +         +         +  2      + 13      +
    .04      4         +         +       2 +     3 1 +         +
    .06      +        4+         + 3       1         +         +
    .08      +         +3     2 4+ 1       +         +         +
    .10      +   3     +  2 1    +      4  +         +         +
    .12      +3       2+         +         +  4      +         +
    .14      +3   1 2  +         +         +      4  +         +
    .16      +31  2    +         +         +         +4        +
    .18      +132 : : :+: : : : :+: : : : :+: : : : :+: 4 : : :+
    .20      1 3       +         +         +         +   4     +
    .22      12 3      +         +         +         +    4    +
    .24      12  3     +         +         +         +     4   +
    .26      21  3     +         +         +         +     4   +
    .28      2 1  3    +         +         +         +      4  +
    .30      2 1  3    +         +         +         +      4  +
    .32      2  1 3    +         +         +         +      4  +
    .34      2   13    +         +         +         +      4  +
    .36      2   13    +         +         +         +      4  +
    .38      2: : 3 : :+: : : : :+: : : : :+: : : : :+: : : 4 :+
    .40      2    3    +         +         +         +      4  +
    .42      2    3    +         +         +         +      4  +
    .44      2    31   +         +         +         +      4  +
    .46      2    31   +         +         +         +      4  +
    .48      2    31   +         +         +         +      4  +
    .50      2    3 1  +         +         +         +      4  +
    .52      2    3 1  +         +         +         +      4  +
    .54      2    3 1  +         +         +         +      4  +
    .56      2    3 1  +         +         +         +      4  +
    .58      2: : 3 :1:+: : : : :+: : : : :+: : : : :+: : : 4 :+
    .60      2    3  1 +         +         +         +      4  +
    .62      2    3  1 +         +         +         +      4  +
    .64      2    3  1 +         +         +         +      4  +
    .66      2    3  1 +         +         +         +      4  +
    .68      2    3  1 +         +         +         +      4  +
    .70      2    3  1 +         +         +         +      4  +
    .72      2    3   1+         +         +         +      4  +
    .74      2    3   1+         +         +         +      4  +
    .76      2    3   1+         +         +         +      4  +
    .78      2: : 3 : 1+: : : : :+: : : : :+: : : : :+: : : 4 :+
    .80      2    3   1+         +         +         +      4  +
    .82      +2   3   1+         +         +         +      4  +
    .84      +2   3   1+         +         +         +      4  +
    .86      +2   3   1+         +         +         +      4  +
    .88      +2   3   1+         +         +         +      4  +
    .90      +2   3   1+         +         +         +      4  +
    .92      +2   3   1+         +         +         +      4  +
    .94      +2   3   1+         +         +         +      4  +
    .96      +2   3   1+         +         +         +      4  +
    .98      +2 : 3 : 1+: : : : :+: : : : :+: : : : :+: : : 4 :+
SCALE 1     -.3000    -.0000     .3000     .6000     .9000    1.2000
SCALE 2     -.1000     .2000     .5000     .8000    1.1000    1.4000
SCALE 3    -3.0000    3.0000    9.0000   15.0000   21.0000   27.0000
SCALE 4   -43.0000  -34.0000  -25.0000  -16.0000   -7.0000    2.0000
```

Figure V-3. PI for QWS-2 (CGTPIF)

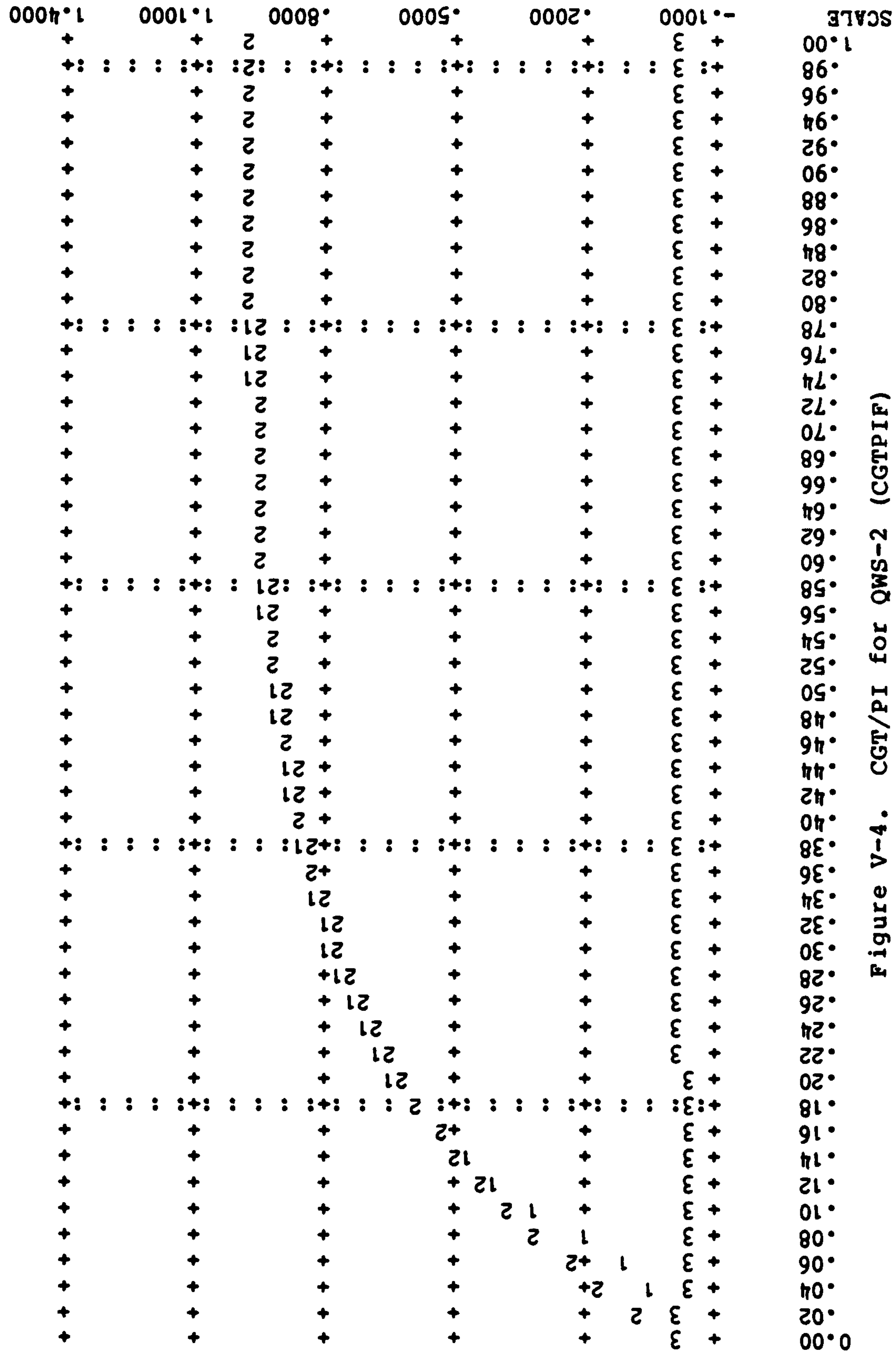


Figure V-4. CGT/PI for QWS-2 (CGTPIF)

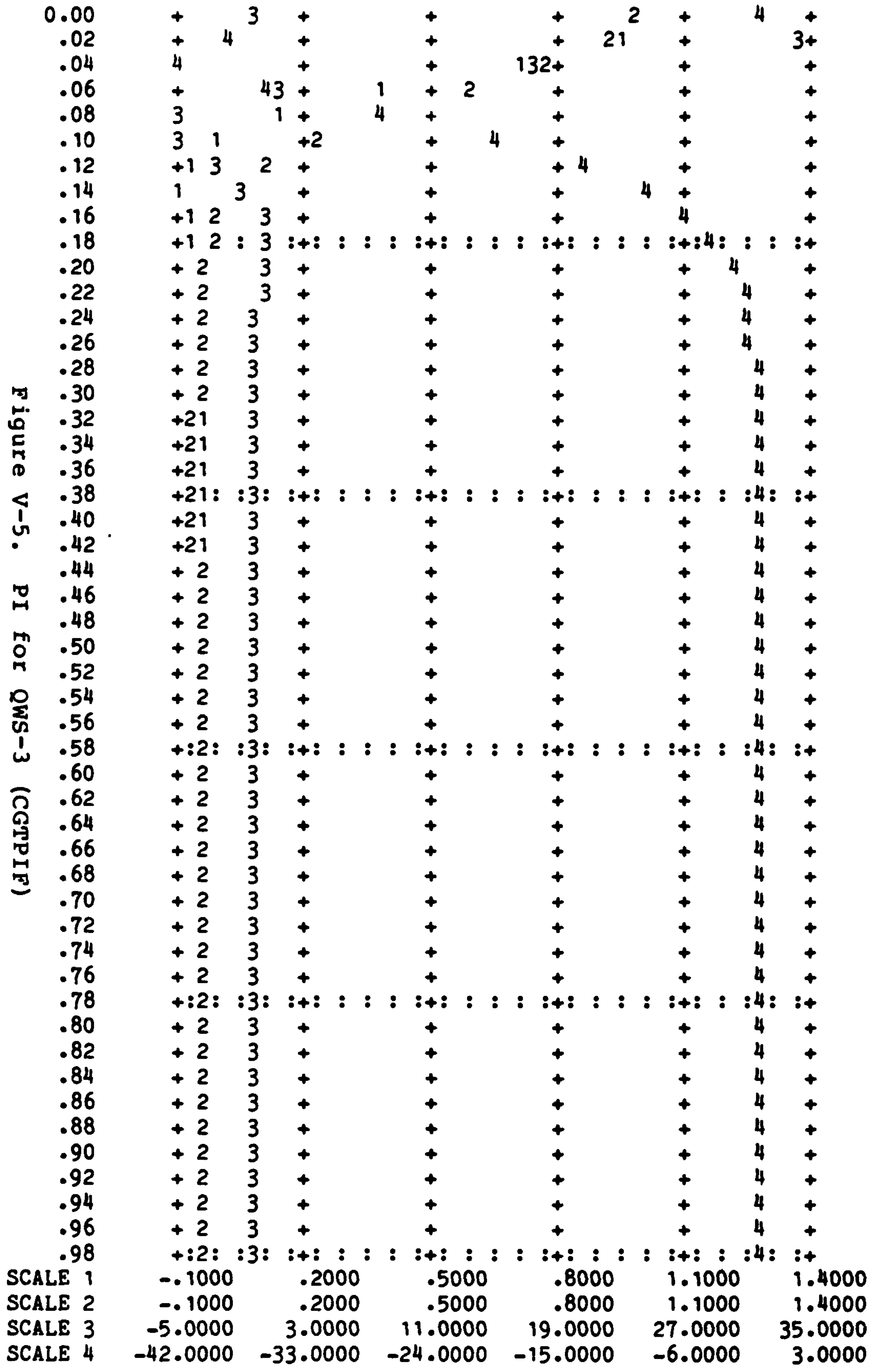


Figure V-5. PI for QWS-3 (CGTPIF)

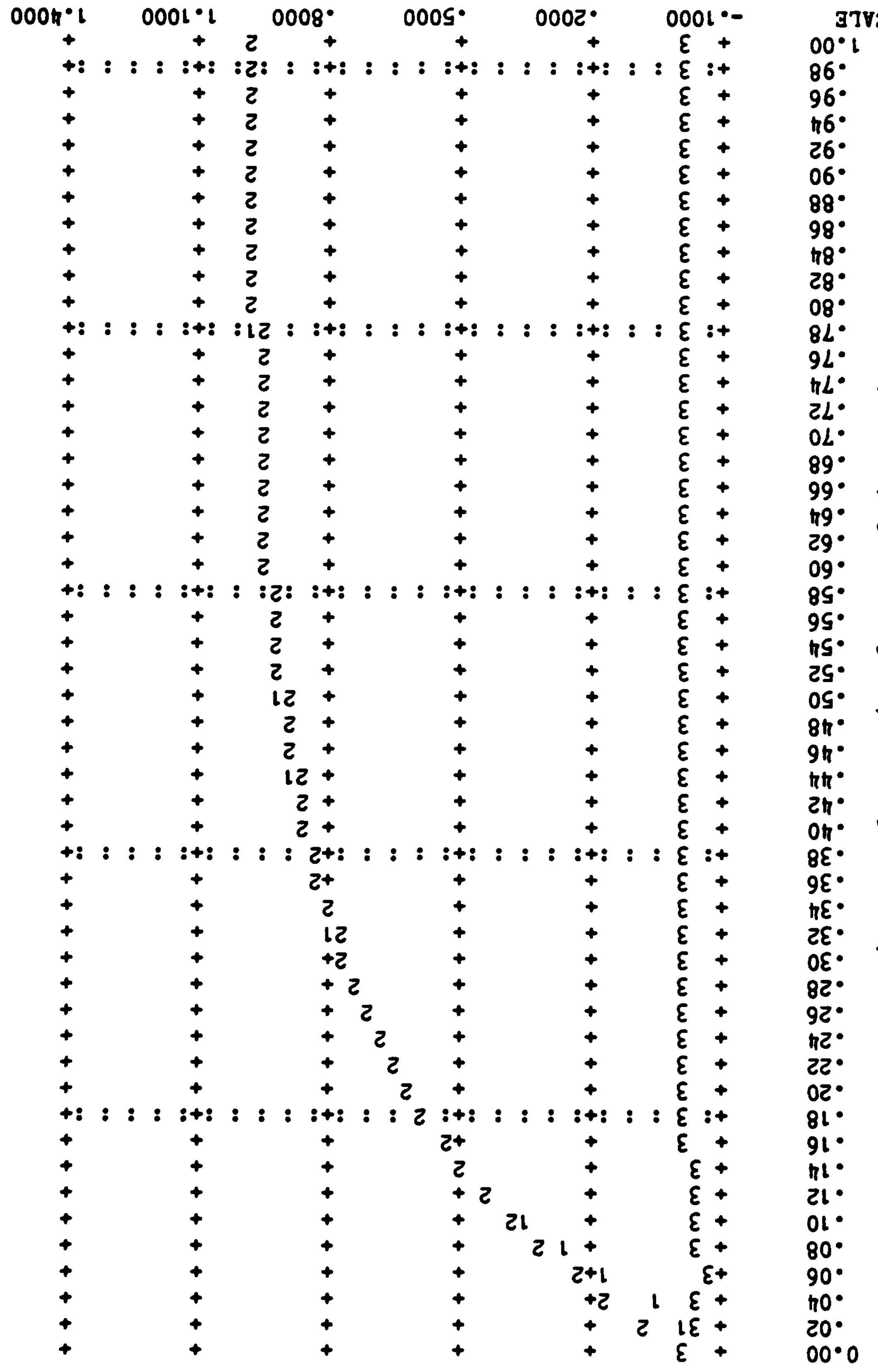


Figure V-6. CGT/PI for QWS-3 (CGTPIF)

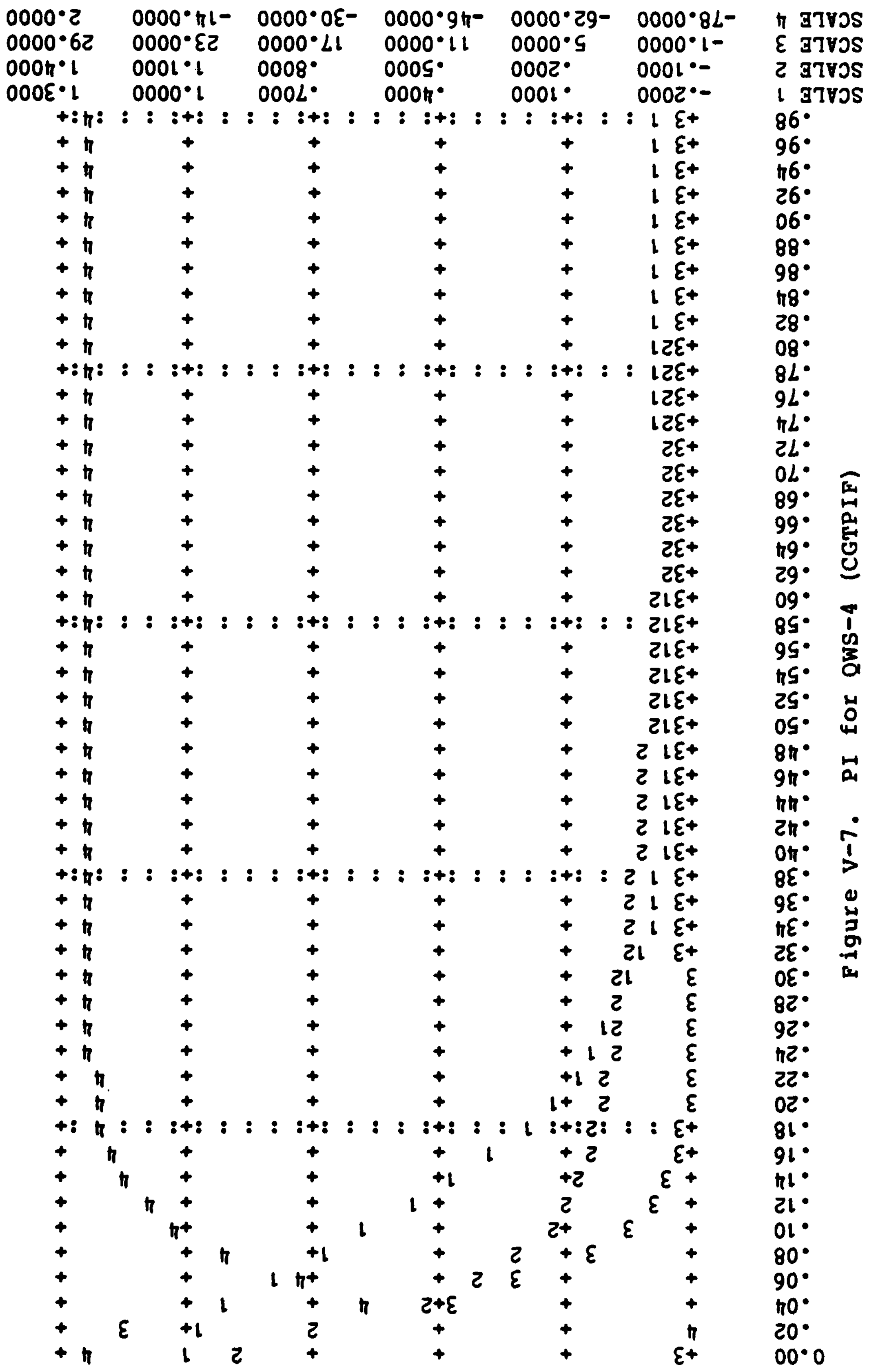


Figure V-7. PI for QWS-4 (CGTPIF)

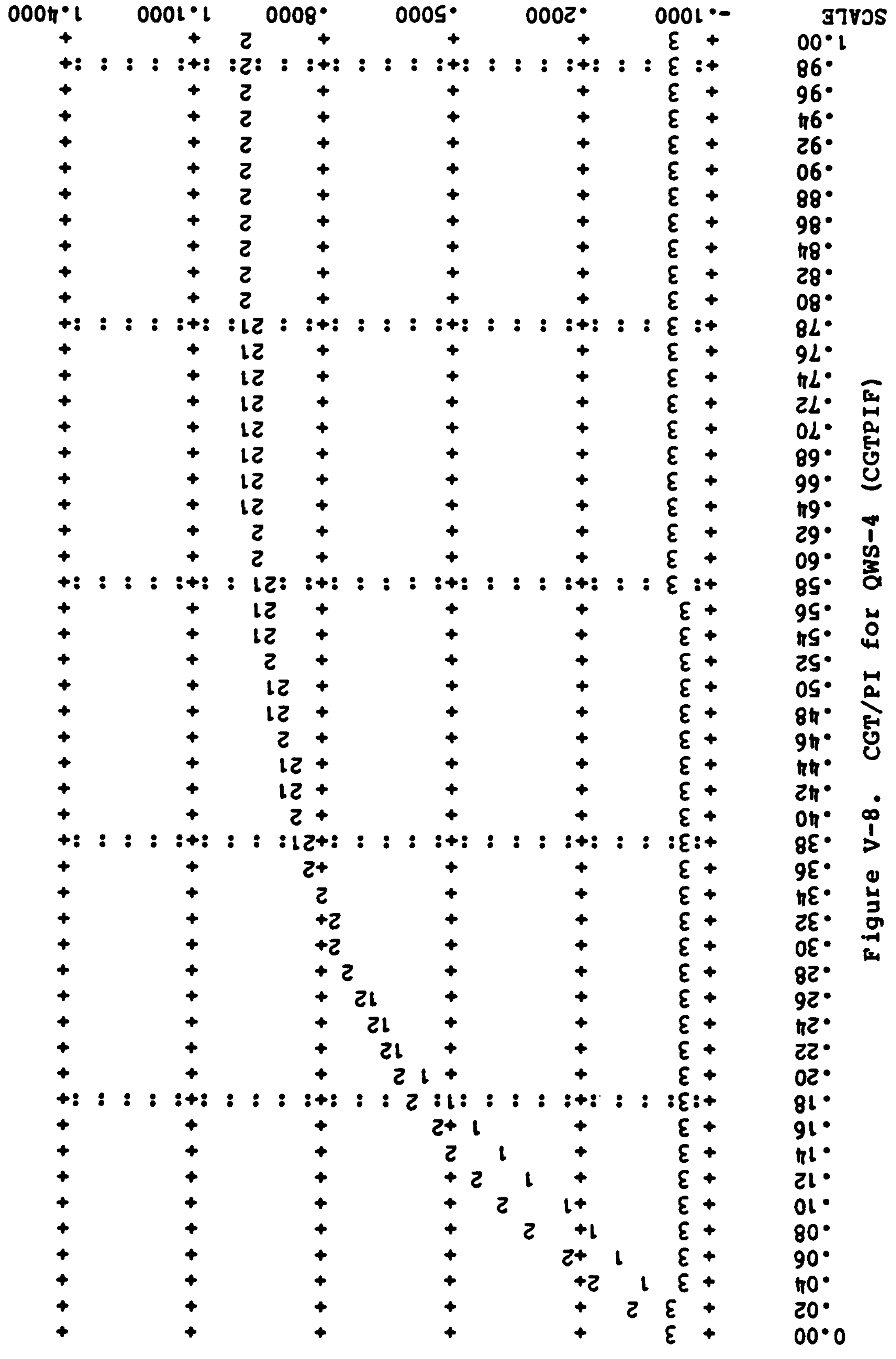


Figure V-8. CGT/PI for QWS-4 (CGTPIF)

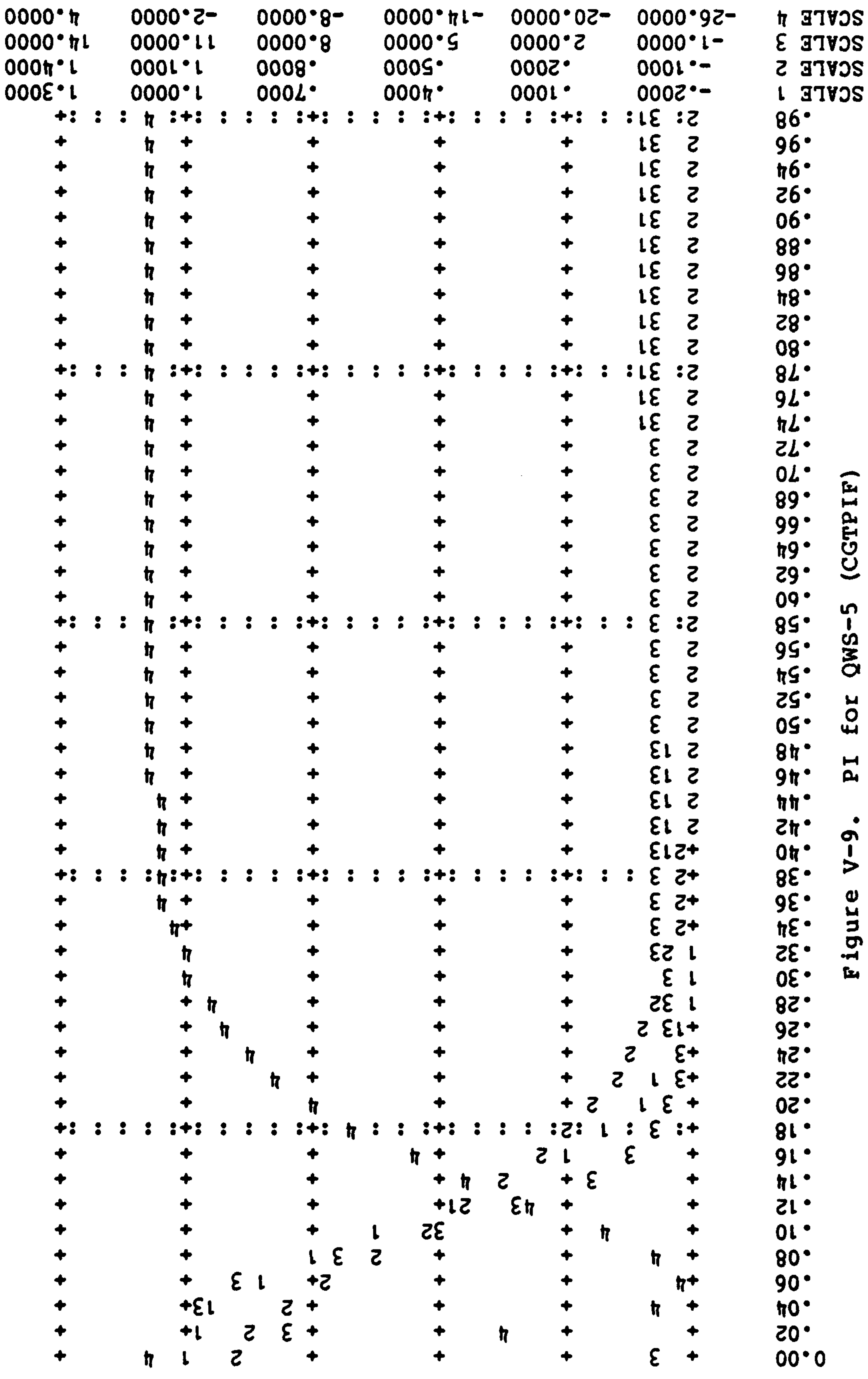


Figure V-9. PI for QWS-5 (CGTPIF)

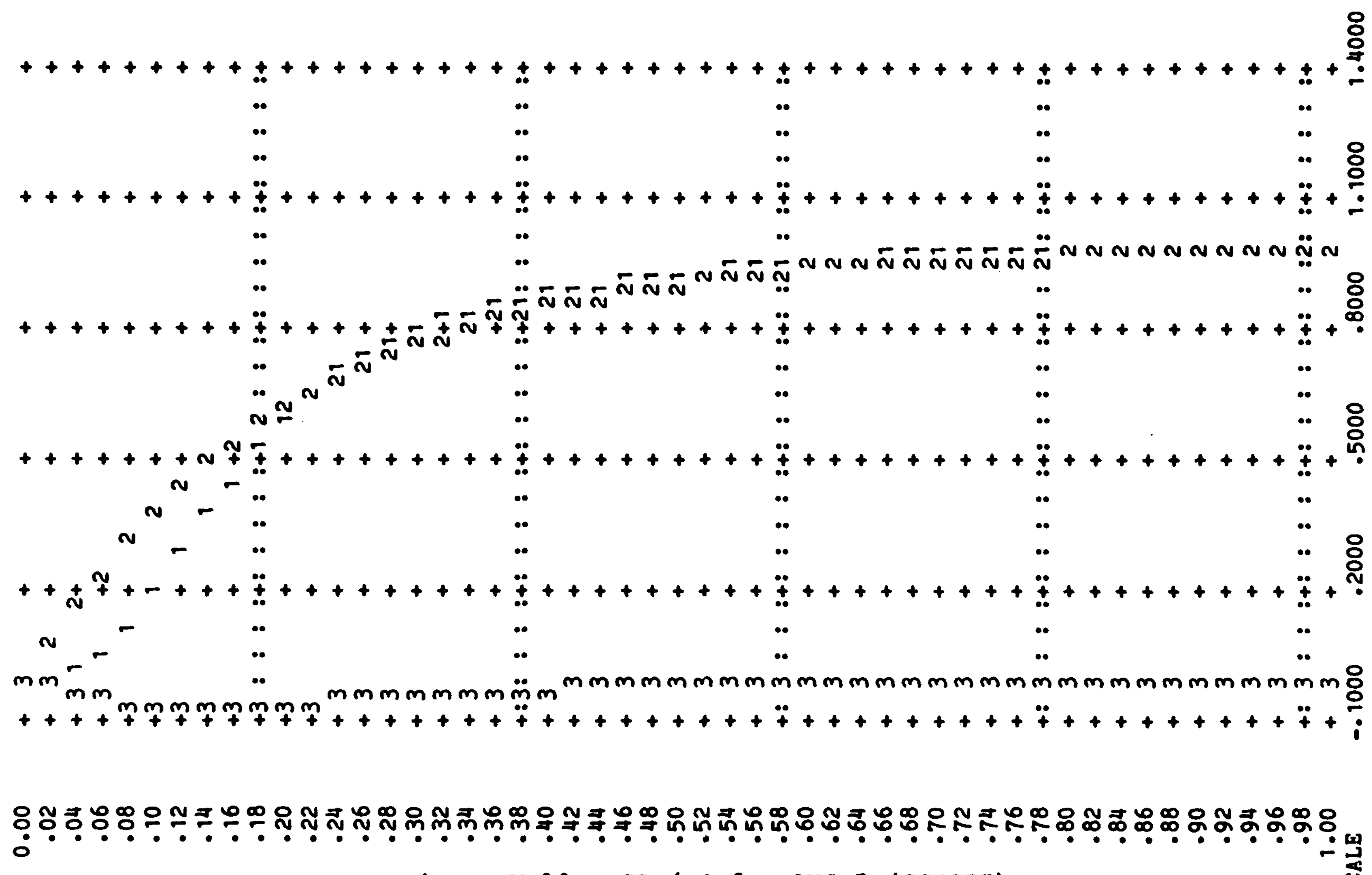


Figure V-10. CGT/PI for QWS-5 (CGTPIF)

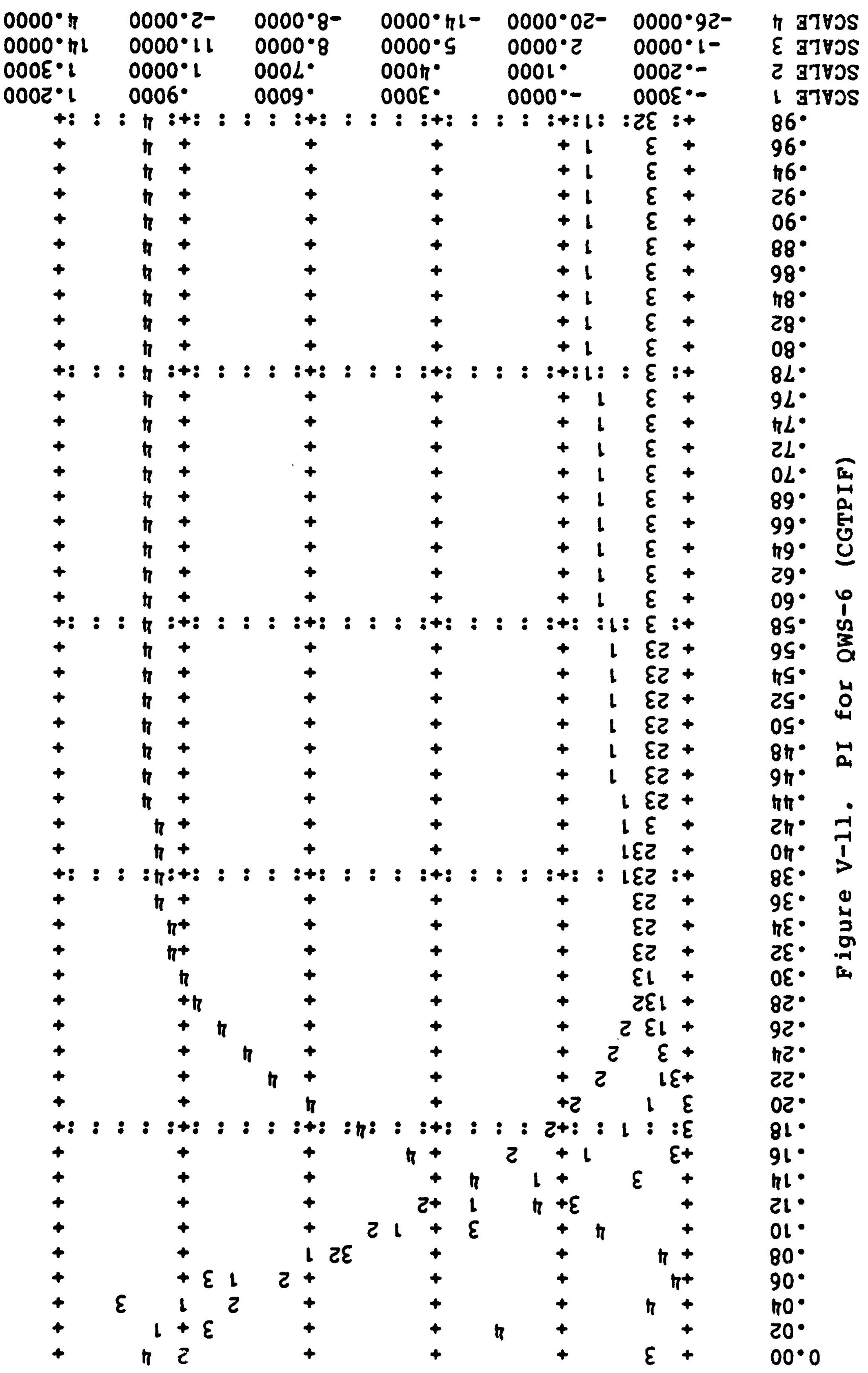


Figure V-11. PI for QWS-6 (CGTPIF)

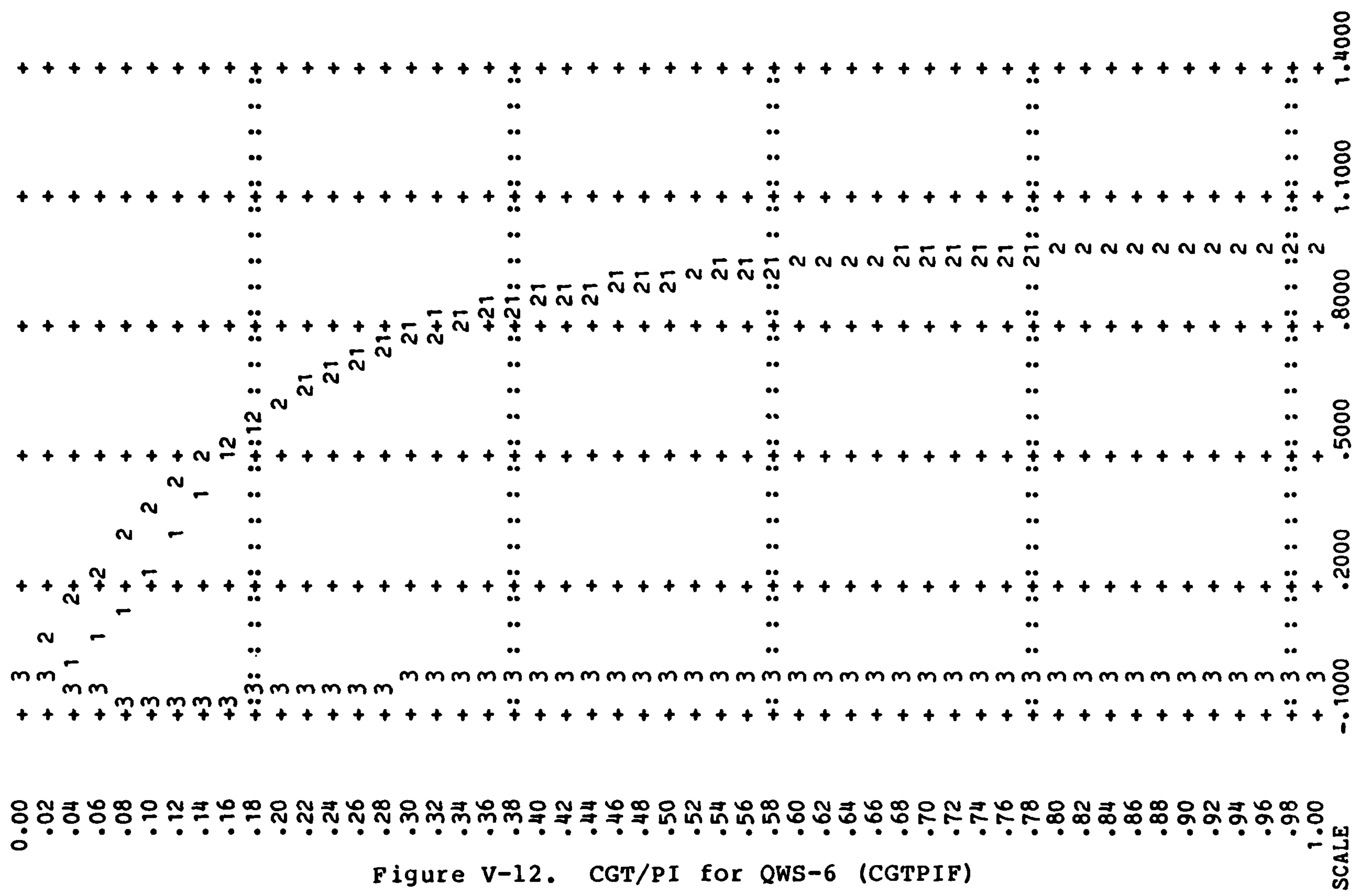


Figure V-12. CGT/PI for QWS-6 (CGTPIF)

Table V-2: CGTPIQ Gains and Pole Positions for Various PI Quadratic Weights

Wts Y;X33;UM;UR	GC1	GC2=KI	PI Poles	KP
1,1;1;1,1;1,1 1	-17.83 5.71 -1.51 1.17 0.21 12.75 -13.9 -0.15 0.20 0.20	-0.65 -0.41 -0.43 0.86	-19.2+j12.4 -4.88+j5.18 -2.73+j2.2, -20	-4.66 -5.46 -7.45 11.67
10,10;1;1,1;1,1 2	-18.16 5.75 -1.53 1.18 0.21 12.93 -14.1 -0.15 -0.2 0.20	-0.65 -0.40 -0.43 0.86	-19.1+j12.5 -5.14+j4.97 -2.76+j2.1, -20	-5.03 -5.51 -7.51 11.95
1,1;10;1,1;1,1 3	-18.84 4.63 -2.49 1.75 0.35 13.75 -14.5 -0.44 0.35 0.24	-0.55 -0.33 -0.33 0.89	-30.6+j26.2 -2.8+j3.1 -2.66+j2.0, -19.9	-5.68 -4.39 -6.96 12.19
1,1;1;10,10;1,1 4	-34.28 10.64 -2.06 1.44 0.22 29.88 -31.88 -0.07 0.21 0.38	-2.04 -0.97 -0.97 2.76	-19.4+j11.9 -4.4+j4.3 -8.3+j9.6, -20.1	-10.5 -8.62 -8.28 21.05
1,1;1;1,1;10,10 5	-8.54 2.98 -0.83 0.74 0.14 4.46 -5.28 -0.10 0.13 0.09	-0.20 -0.2 -0.19 0.24	-4.0+j5.2 -1.2+j65.9 -11,-17.5, -20	-1.29 -3.69 -7.3 7.84
5,5;2;1,1;10,10 6	-34.59 10.36 -2.22 1.53 0.25 30.12 -32.04 -0.11 0.23 0.38	-1.98 -0.9 -0.93 2.77	-21.8+j16.0 -7.37+j7.8 -4.4+j4.3, -20.1	-10.87 -8.32 -8.18 21.25
5,5;20;5,5;1,1 X44=1,X55=1 7	-29.43 7.77 -3.13 2.07 0.35 24.63 -26.0 -0.53 0.36 0.61	-1.08 -0.6 -0.60 1.88	-36.5+j30 -3.5+j3.8 -3.3+j2.7, -28.3	-9.13 -6.62 -9.26 19.41
Ex:5,5;5,5;1,1 Im:2,2;2,2 8	-20.97 6.63 -1.57 1.20 0.20 16.12 -17.2 -0.10 0.19 0.25	-0.85 -0.5 -0.51 1.12	-17.8+j10.4 -6.6+j6.1 -3.04+J2.5, -20.4	-6.19 -6.15 -7.58 13.64

Table V-2 continued: CGTPIQ Gains and Pole Positions For Various Quadratic Weights

	KXM-1	KXU-1	KXM-2	KXU-2	KXM-3,4	KXU-3,4
1	-0.54 -0.81 -4.19 -5.51	-4.12 -4.65 -3.26 17.18	-4.31 -3.18 -5.92 2.69	-0.36 -2.28 -1.52 8.98	-4.31 -3.18 -5.92 2.69	-5.25 -5.09 -3.16 19.14
2	-0.80 -0.85 -4.24 -5.29	-4.23 -4.66 -3.27 17.24	-4.51 -3.21 -5.95 2.86	-0.51 -2.31 -1.56 9.09	-4.51 -3.21 -5.95 2.86	-5.33 -5.10 -3.17 19.18
3	2.96 -0.04 -2.25 -5.05	-8.64 -4.34 -4.71 17.24	-1.95 -2.39 -4.09 3.17	-3.73 -2.00 -2.87 9.02	-1.95 -2.39 -4.09 3.17	-9.66 -4.84 -4.53 19.19
4	-4.67 -3.27 -5.18 1.77	-5.86 -5.35 -3.10 19.28	-13.0 -7.58 -8.21 18.43	2.44 -1.04 -0.07 2.62	-13.0 -7.58 -8.21 18.43	-7.73 -5.99 -2.88 22.53
5	0.25 0.55 -4.35 -8.44	-1.55 -4.25 -2.95 16.28	-1.06 -0.72 -5.16 -4.73	-0.2 -2.97 -2.14 12.6	-1.06 -0.72 -5.16 -4.73	-2.24 -4.51 -2.93 17.28
6	-4.24 -3.04 -4.87 1.93	-6.63 -5.27 -3.32 19.32	-12.8 -7.35 -7.94 18.57	1.96 -0.96 -0.2 2.68	-12.8 -7.35 -7.94 18.57	-8.47 -5.94 -3.07 22.55
7	2.02 -0.77 -3.70 -4.61	-11.2 -5.85 -5.57 24.02	-5.88 -4.58 -6.56 10.05	-3.2 -2.04 -2.71 9.36	-5.88 -4.58 -6.56 10.05	-12.5 -6.42 -5.20 26.34
8	-19.0 -1.33 -4.50 -4.28	-4.29 -4.82 -3.08 17.91	-6.19 -3.97 -6.37 5.35	0.01 -2.17 -1.2 8.29	-6.19 -3.97 -6.37 5.35	-5.52 -5.29 -2.95 20.08

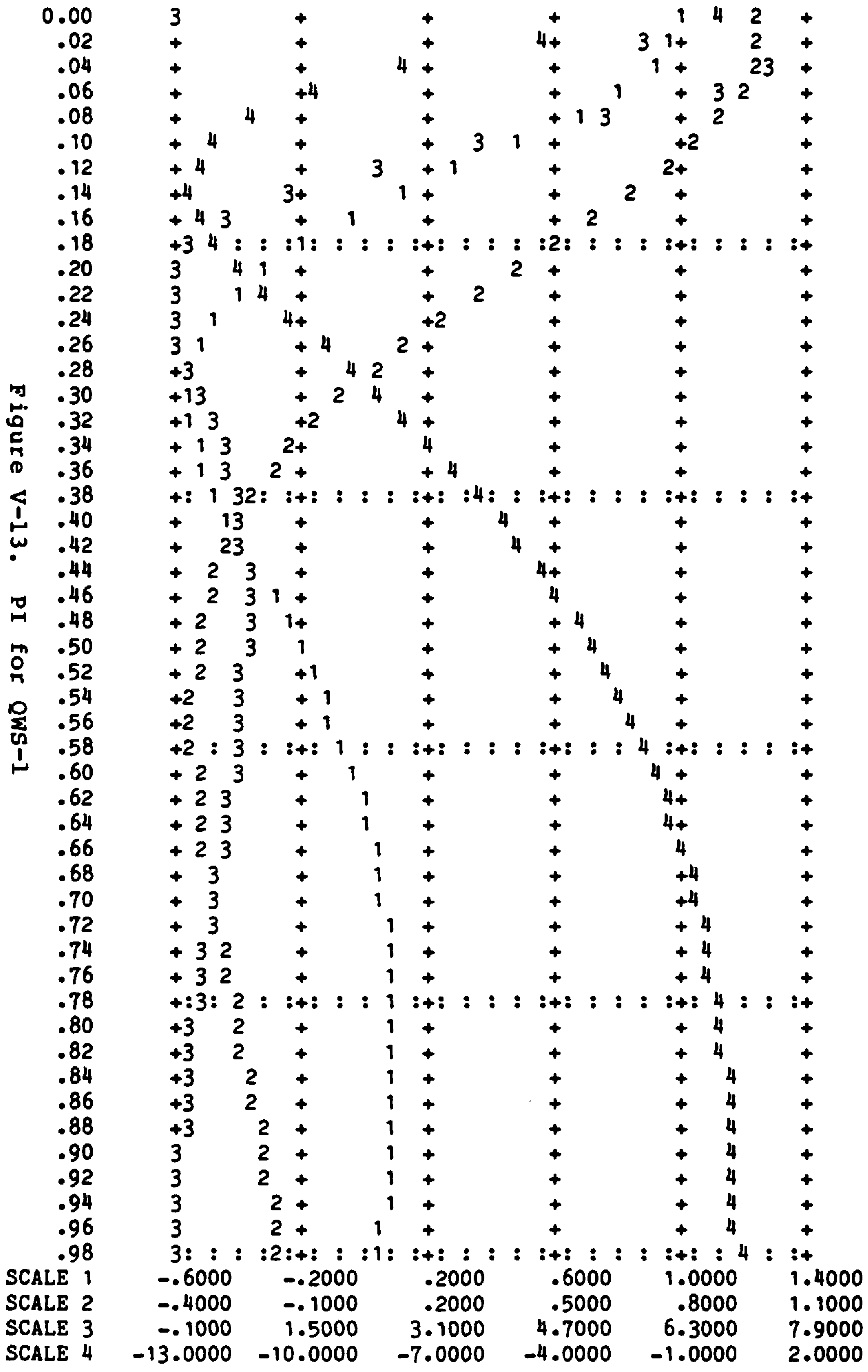


Figure V-13. PI for QWS-1

```
 0.00  +  3      +         +         +         +         +
  .02  +  3  2   +         +         +         +         +
  .04  + 3 1    2+         +         +         +         +
  .06  + 3  1    +2        +         +         +         +
  .08  +3    1   +   2     +         +         +         +
  .10  +3      1 +     2   +         +         +         +
  .12  +3        1       2 +         +         +         +
  .14  +3        + 1       2         +         +         +
  .16  +3        +   1     +2        +         +         +
  .18  +3 : : : :+: : :1: :+: 2 : : :+: : : : :+: : : : :+
  .20  +3        +       1 +   2     +         +         +
  .22  +3        +        1+    2    +         +         +
  .24  +3        +         1     2   +         +         +
  .26  +3        +         +1     2  +         +         +
  .28  +3        +         + 1     2 +         +         +
  .30  +3        +         +  1     2+         +         +
  .32  +3        +         +  1     2+         +         +
  .34  +3        +         +   1     2         +         +
  .36  +3        +         +    1    +2        +         +
  .38  +3 : : : :+: : : : :+: : 1 : :+2 : : : :+: : : : :+
  .40  +3        +         +     1   + 2       +         +
  .42  +3        +         +     1   + 2       +         +
  .44  +3        +         +      1  + 2       +         +
  .46  +3        +         +      1  +  2      +         +
  .48  +3        +         +       1 +  2      +         +
  .50  +3        +         +       1 +  2      +         +
  .52  +3        +         +        1+   2     +         +
  .54  +3        +         +        1+   2     +         +
  .56  +3        +         +         1   2     +         +
  .58  +:3: : : :+: : : : :+: : : : :1: :2: : :+: : : : :+
  .60  + 3       +         +         1    2    +         +
  .62  + 3       +         +         +1   2    +         +
  .64  + 3       +         +         +1   2    +         +
  .66  + 3       +         +         + 1  2    +         +
  .68  + 3       +         +         + 1  2    +         +
  .70  + 3       +         +         + 1  2    +         +
  .72  + 3       +         +         +  1 2    +         +
  .74  + 3       +         +         +  1 2    +         +
  .76  + 3       +         +         +  1 2    +         +
  .78  +:3: : : :+: : : : :+: : : : :+: 1 2 : :+: : : : :+
  .80  + 3       +         +         +   1 2   +         +
  .82  + 3       +         +         +   1 2   +         +
  .84  + 3       +         +         +   1 2   +         +
  .86  + 3       +         +         +   1 2   +         +
  .88  +  3      +         +         +    12   +         +
  .90  +  3      +         +         +    12   +         +
  .92  +  3      +         +         +    12   +         +
  .94  +  3      +         +         +    12   +         +
  .96  +  3      +         +         +    12   +         +
  .98  +: 3 : : :+: : : : :+: : : : :+: : 12: : :+: : : : :+
 1.00  +  3      +         +         +    12   +         +
SCALE -.1000    .2000     .5000     .8000    1.1000    1.4000
```

Figure V-14. CGT/PI-1 for QWS-1

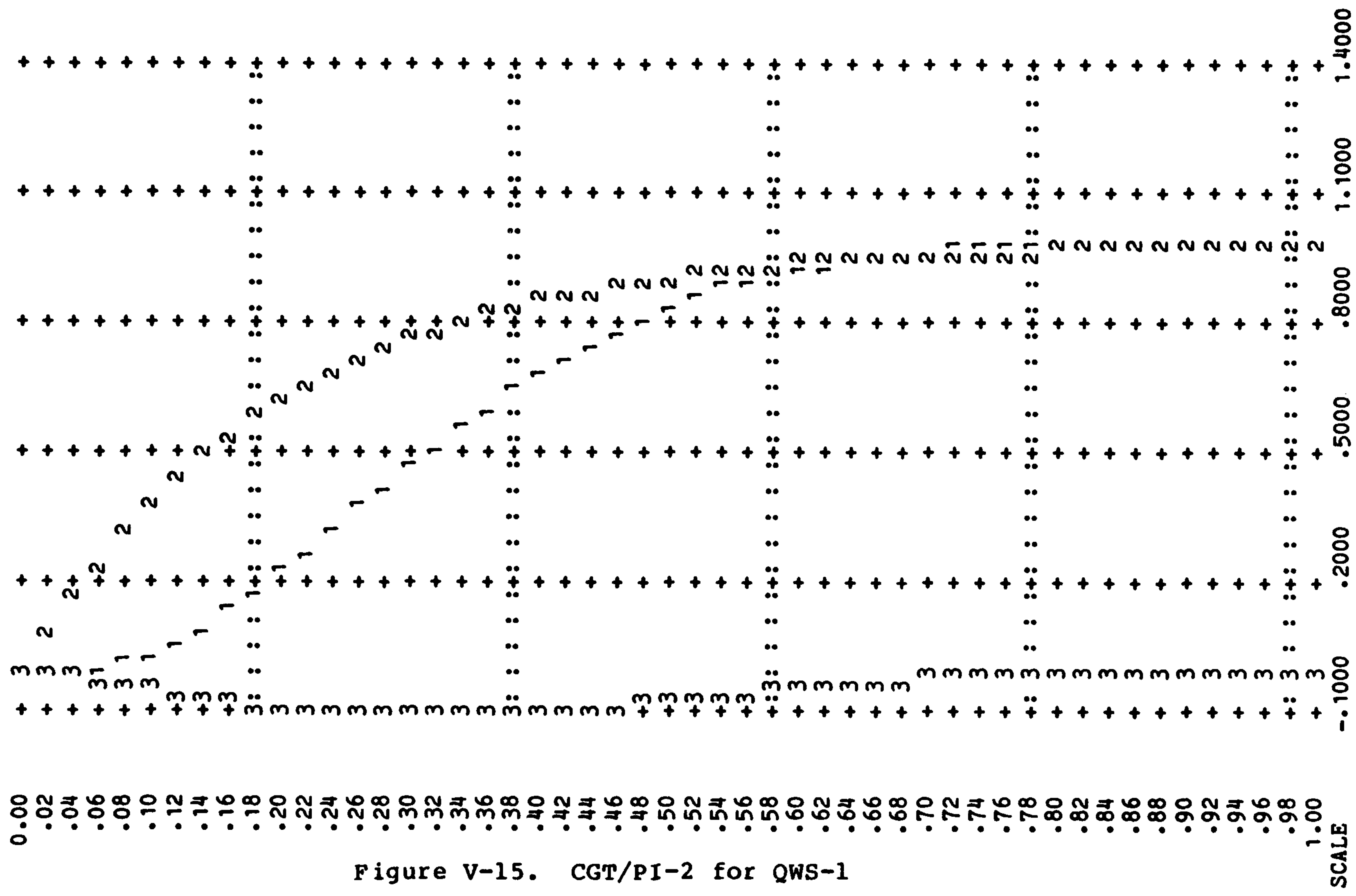


Figure V-15. CGT/PI-2 for QWS-1

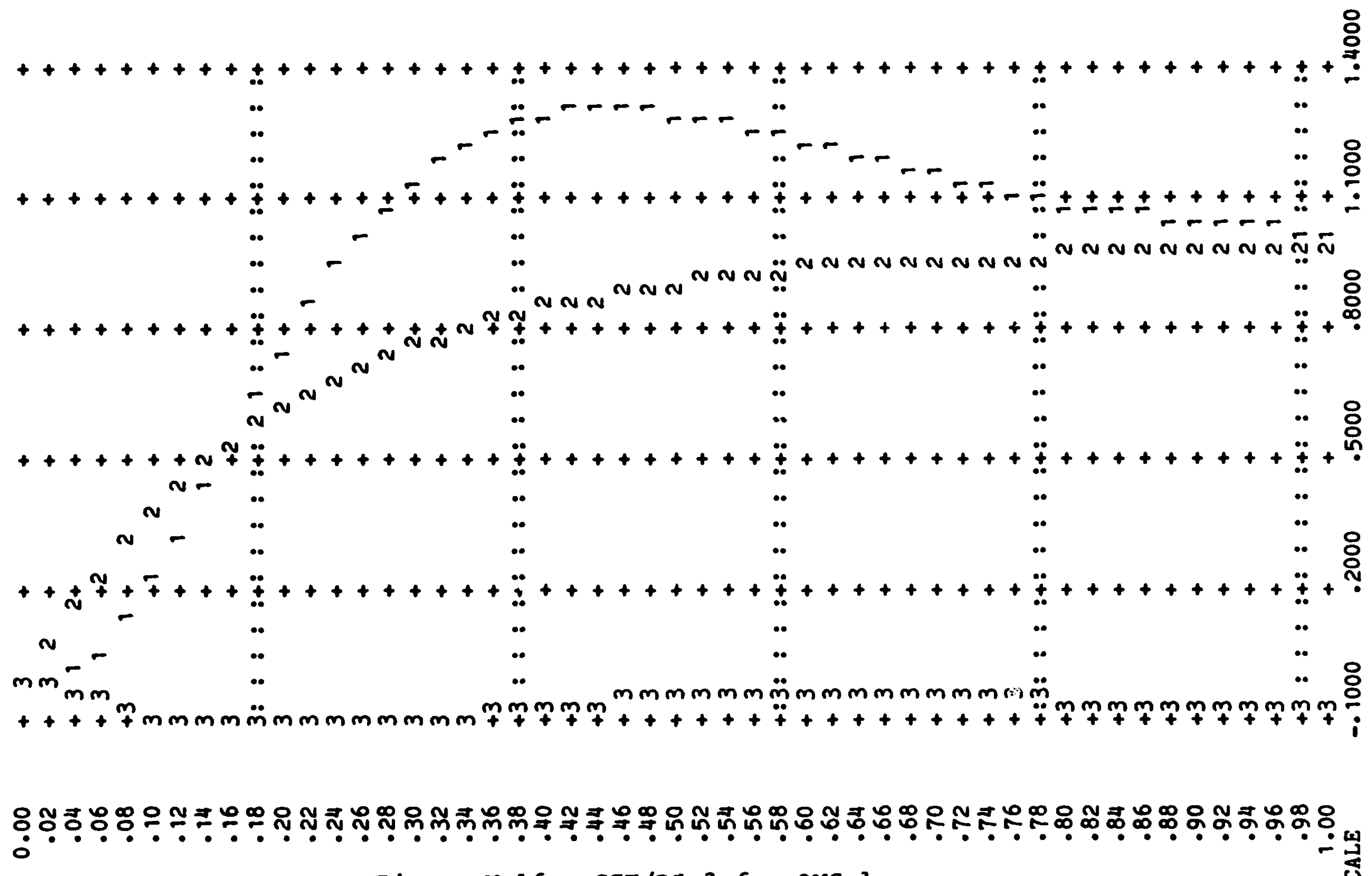


Figure V-16. CGT/PI-3 for QWS-1

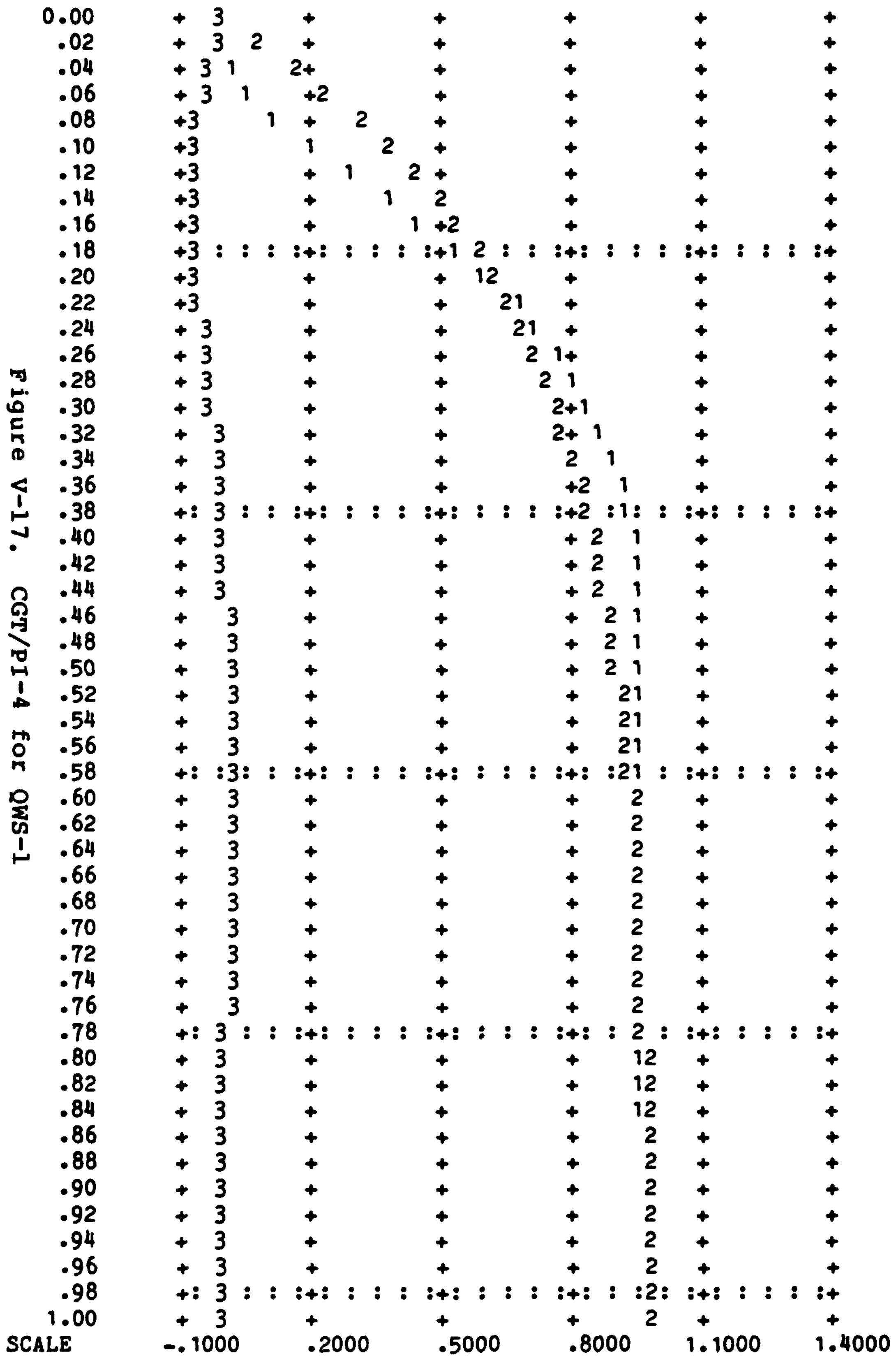


Figure V-17. CGT/PI-4 for QWS-1

```
    0.00      3         +         +         +         1  4  2   +
     .02      +         +         +        4+        31+     2   +
     .04      +         +       4 +         +       1 +     2 3 +
     .06      +         +4        +         +    1    +   32    +
     .08      +     4   +         +         +1  3     +  2      +
     .10      +  4      +         +   3  1  +         +2        +
     .12      +4        +    3    + 1       +        2+         +
     .14      +4      3 +       1 +         +     2   +         +
     .16      + 4 3     +  1      +         +  2      +         +
     .18      +3 4 : : :1: : : : :+: : : : :2: : : : :+: : : : :+
     .20      3    4 1  +         +     2   +         +         +
     .22      3    1 4  +         +  2      +         +         +
     .24      3  1     4+         +2        +         +         +
     .26      +31       + 4     2 +         +         +         +
     .28      +31       +   42    +         +         +         +
     .30      +13       +  2  4   +         +         +         +
     .32      + 13      +2      4 +         +         +         +
     .34      + 1 3    2+         4         +         +         +
     .36      +  1 3  2 +         + 4       +         +         +
     .38      +: :132: :+: : : : :+: :4: : :+: : : : :+: : : : :+
     .40      +    3    +         +     4   +         +         +
     .42      +   2 3   +         +       4 +         +         +
     .44      +  2  31  +         +        4+         +         +
     .46      +  2  3 1 +         +         +4        +         +
     .48      + 2   3  1+         +         + 4       +         +
     .50      + 2   3   1         +         +  4      +         +
     .52      + 2  3    +1        +         +   4     +         +
     .54      + 2  3    + 1       +         +    4    +         +
     .56      + 2  3    +  1      +         +     4   +         +
     .58      +:2: 3 : :+: 1 : : :+: : : : :+: : : 4 :+: : : : :+
     .60      + 2 3     +   1     +         +       4 +         +
     .62      + 2 3     +   1     +         +        4+         +
     .64      + 2 3     +    1    +         +         4         +
     .66      + 23      +    1    +         +         4         +
     .68      +  3      +     1   +         +         +4        +
     .70      +  3      +     1   +         +         +4        +
     .72      + 32      +     1   +         +         + 4       +
     .74      + 3 2     +     1   +         +         + 4       +
     .76      + 3 2     +      1  +         +         + 4       +
     .78      +:3: 2 : :+: : : 1 :+: : : : :+: : : : :+: 4 : : :+
     .80      +3   2    +      1  +         +         +  4      +
     .82      +3   2    +      1  +         +         +  4      +
     .84      +3    2   +      1  +         +         +   4     +
     .86      +3    2   +      1  +         +         +   4     +
     .88      3      2  +      1  +         +         +   4     +
     .90      3      2  +     1   +         +         +   4     +
     .92      3      2  +     1   +         +         +   4     +
     .94      3       2 +     1   +         +         +   4     +
     .96      3       2 +     1   +         +         +   4     +
     .98      3: : : :2:+: : :1: :+: : : : :+: : : : :+: : 4 : :+
SCALE 1      -.6000    -.2000     .2000     .6000    1.0000    1.4000
SCALE 2      -.4000    -.1000     .2000     .5000     .8000    1.1000
SCALE 3      -.1000    1.5000    3.1000    4.7000    6.3000    7.9000
SCALE 4    -13.0000  -10.0000   -7.0000   -4.0000   -1.0000    2.0000
```

Figure V-18. PI for QWS-2

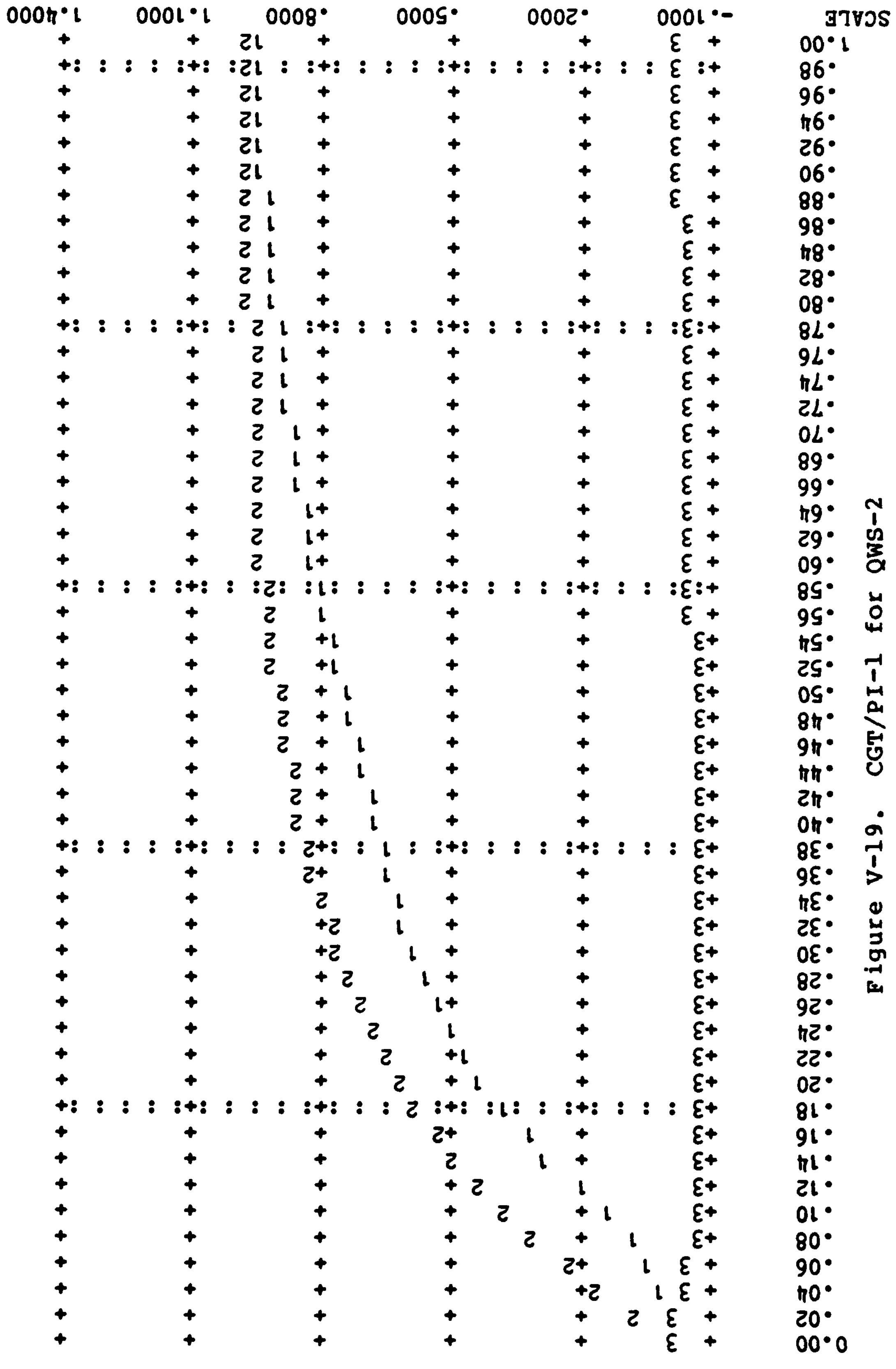


Figure V-19. CGT/PI-1 for QWS-2

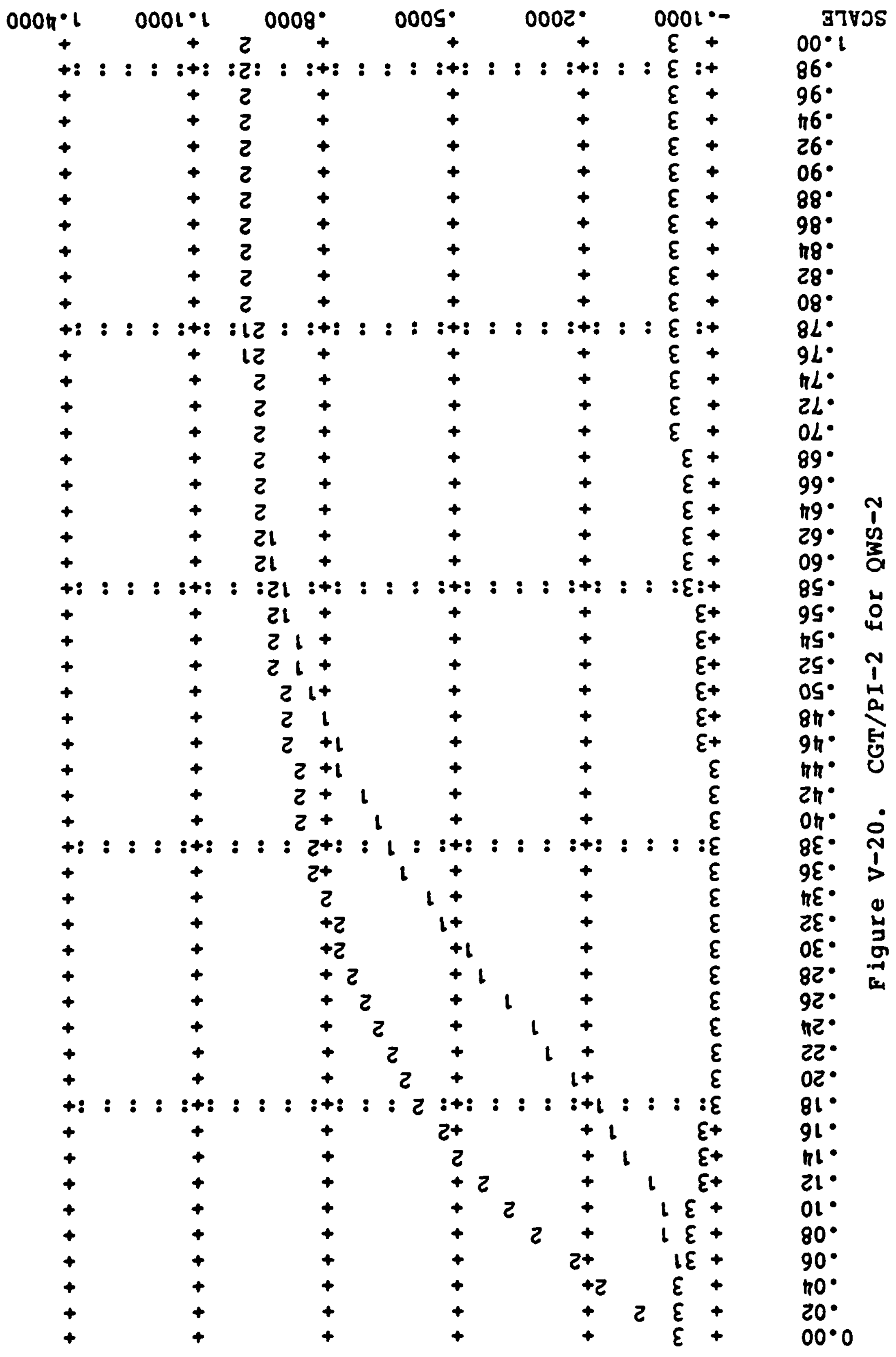


Figure V-20. CGT/PI-2 for QWS-2

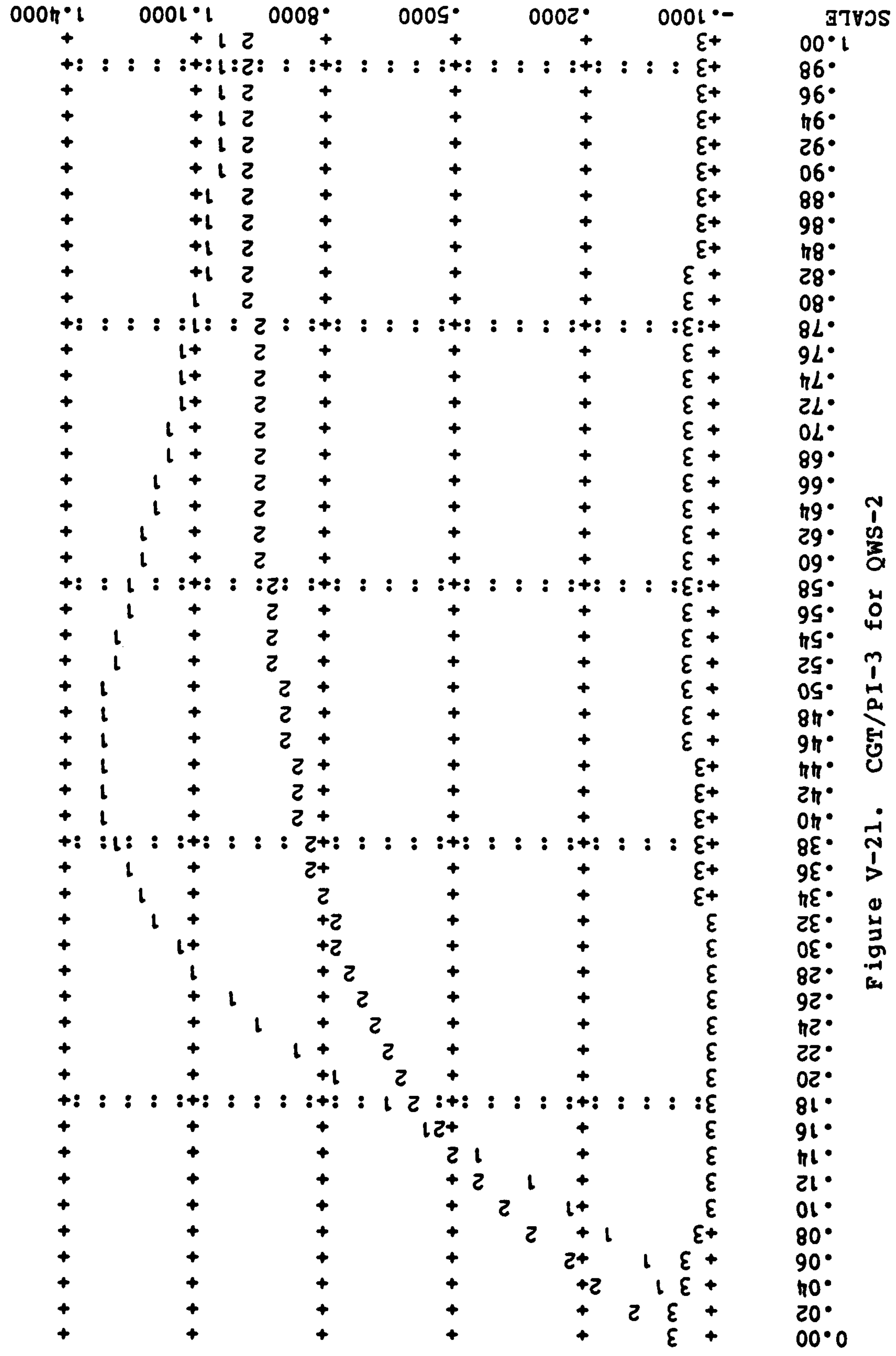


Figure V-21. CGT/PI-3 for QWS-2

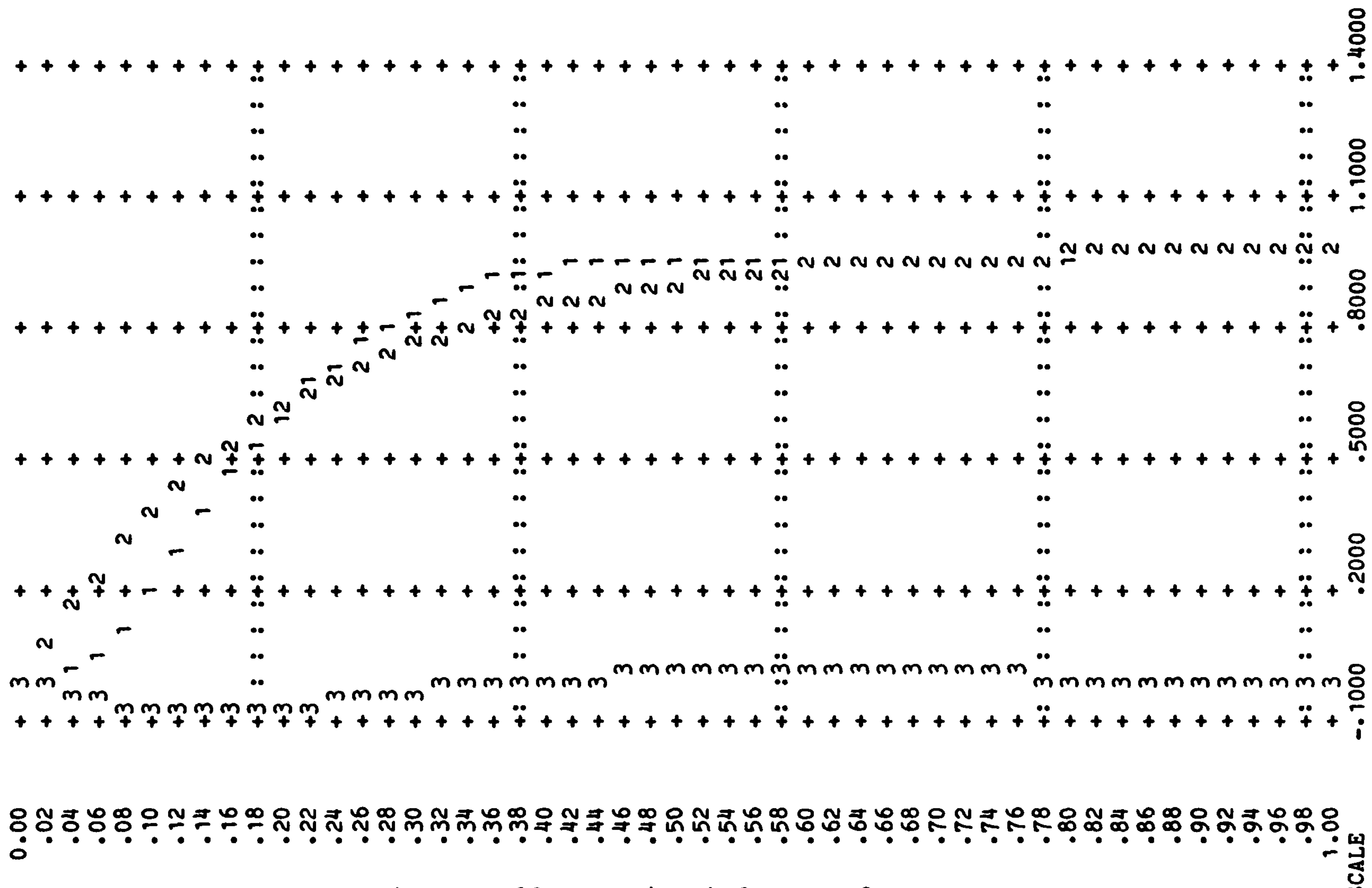


Figure V-22. CGT/PI-4 for QWS-2

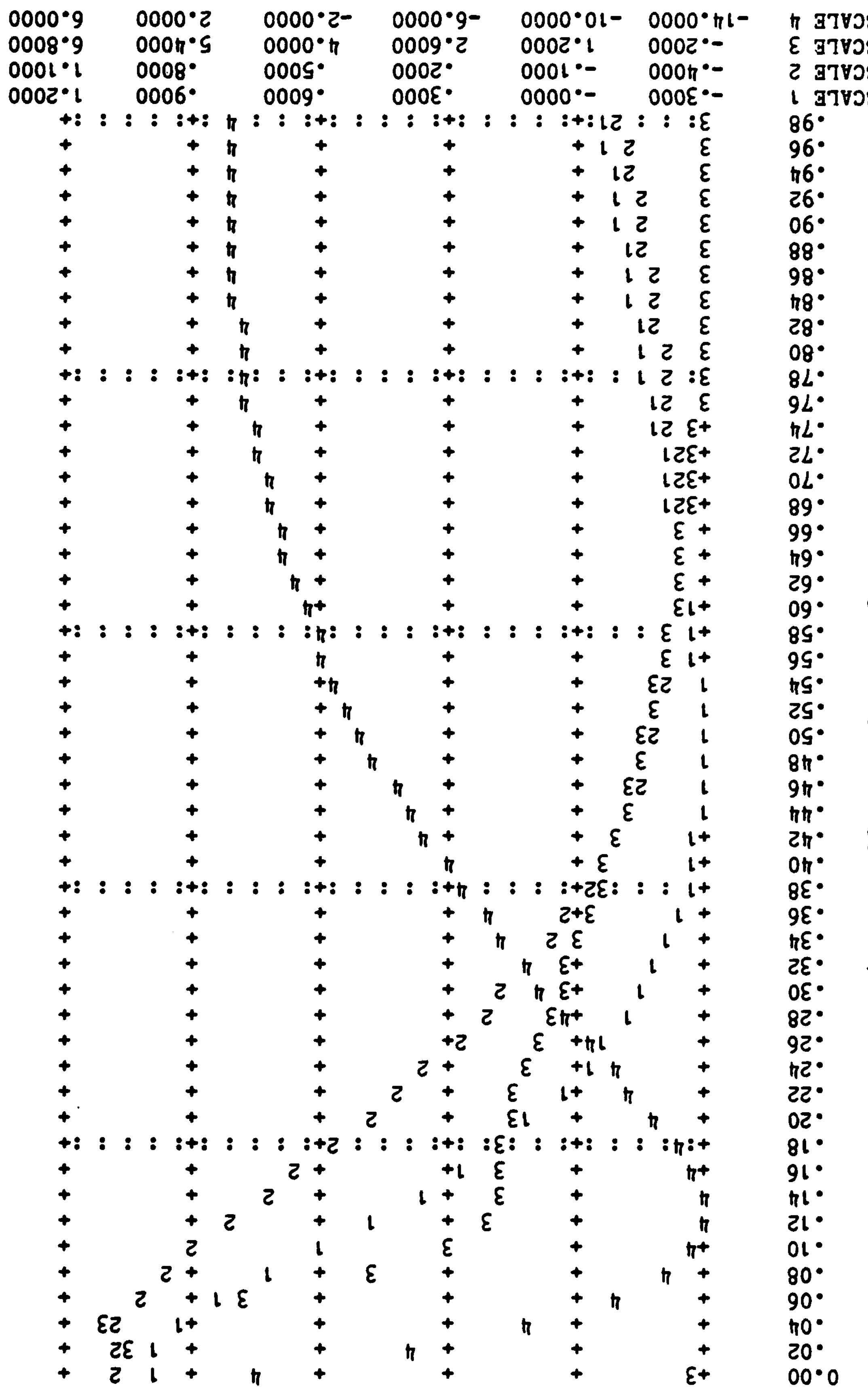


Figure V-23. PI for QWS-3

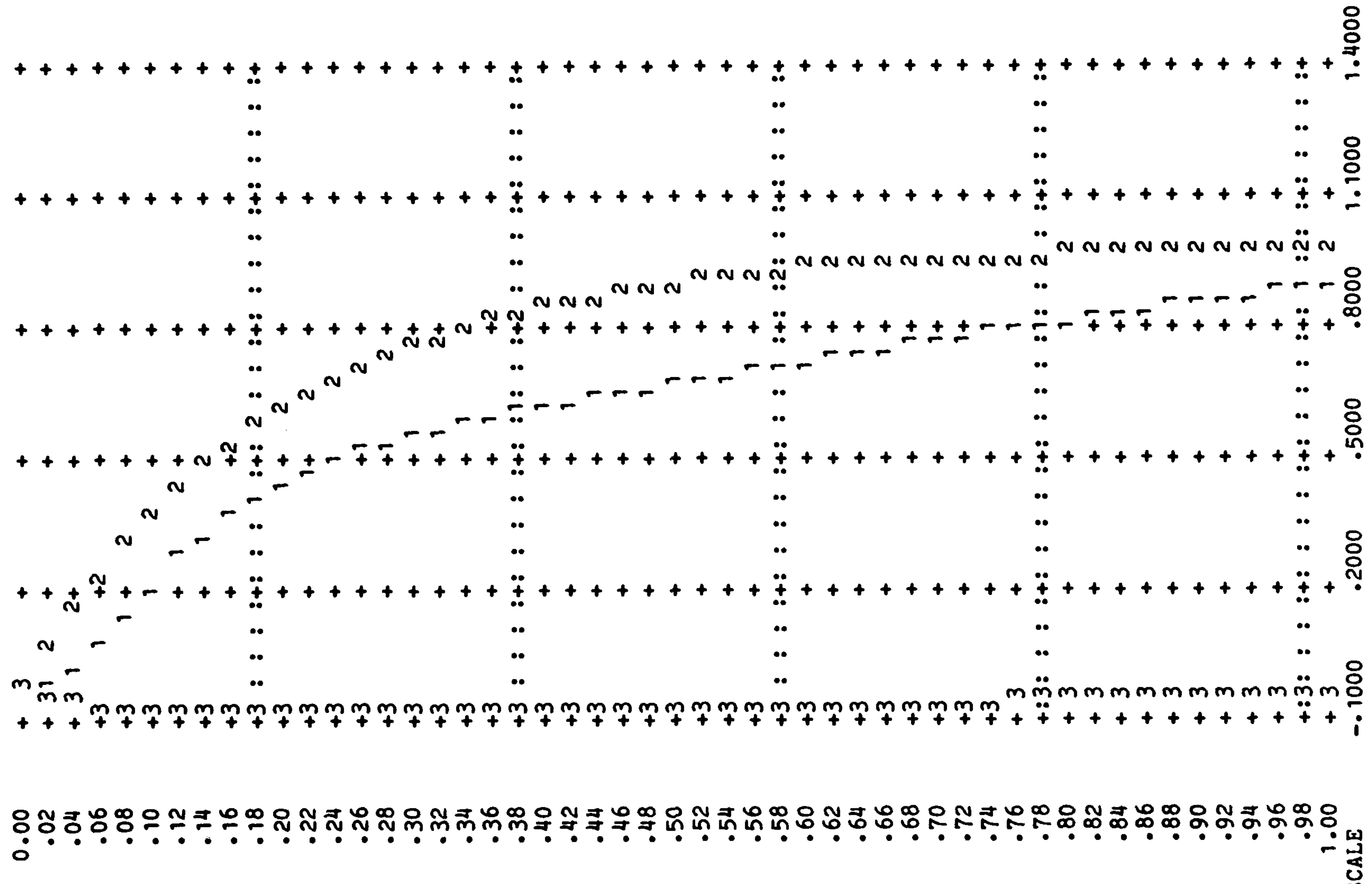


Figure V-24. CGT/PI-1 for QWS-3

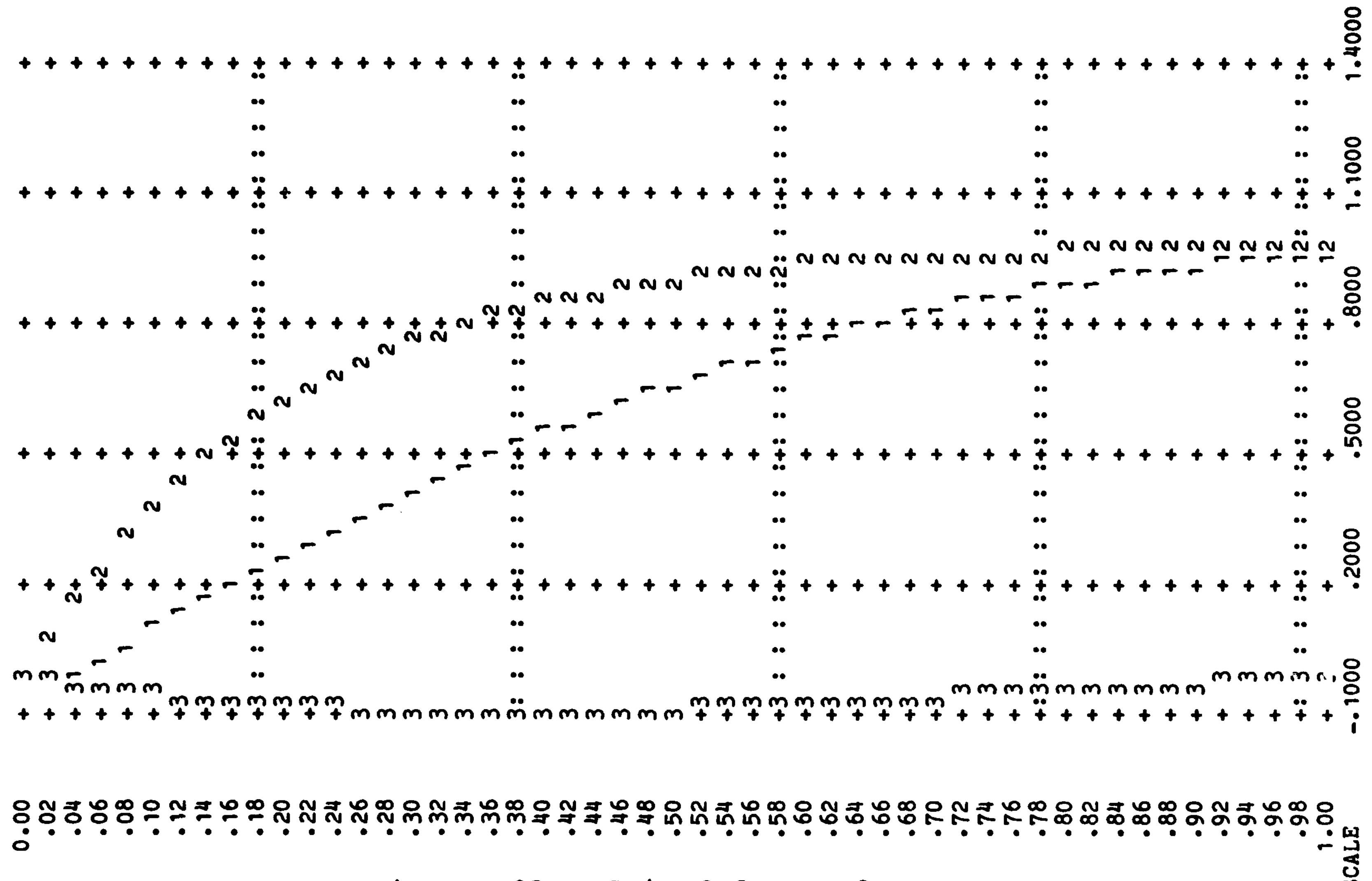


Figure V-25. CGT/PI-2 for QWS-3

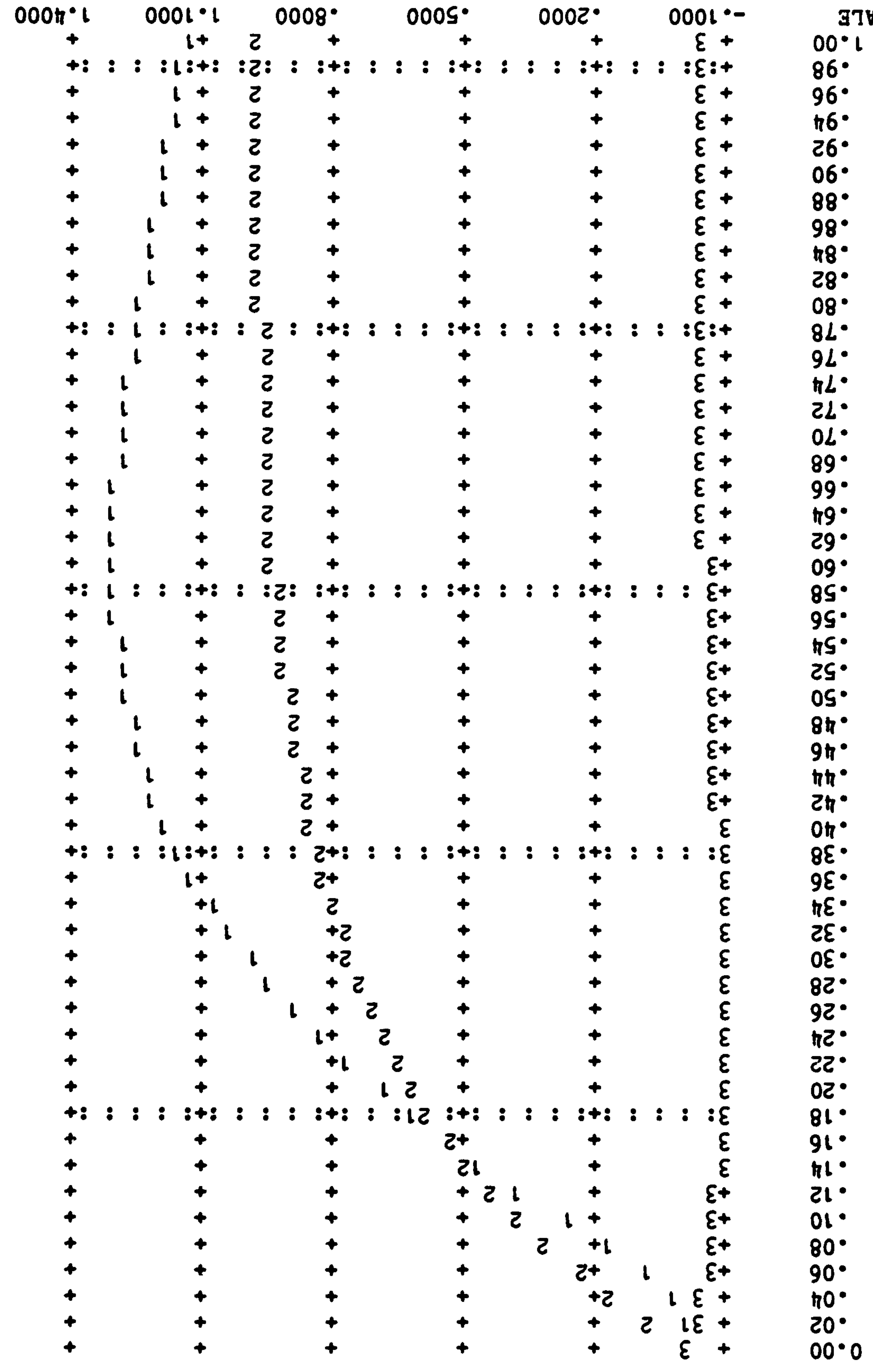


Figure V-26. CGT/PI-3 for QWS-3

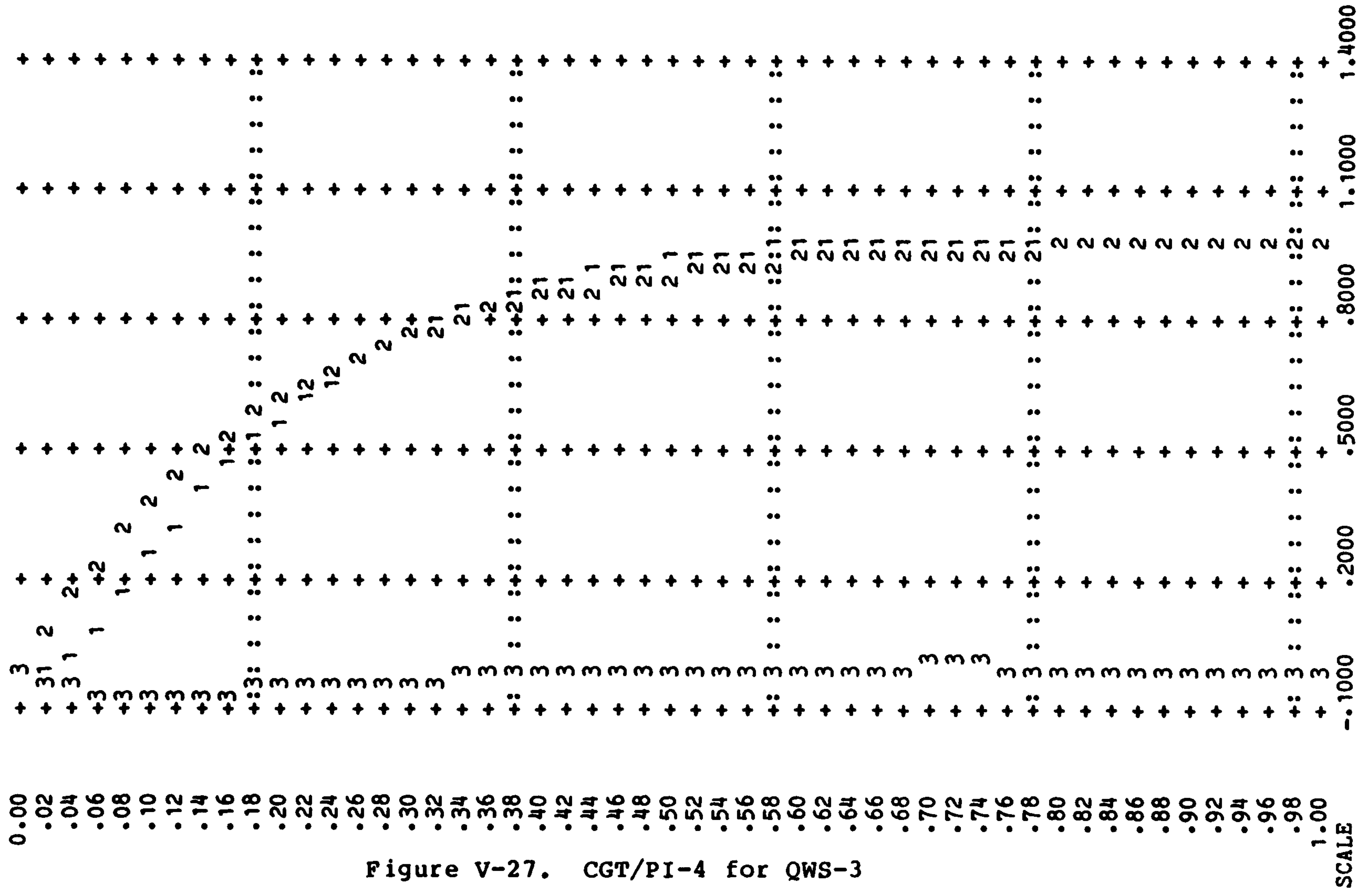


Figure V-27. CGT/PI-4 for QWS-3

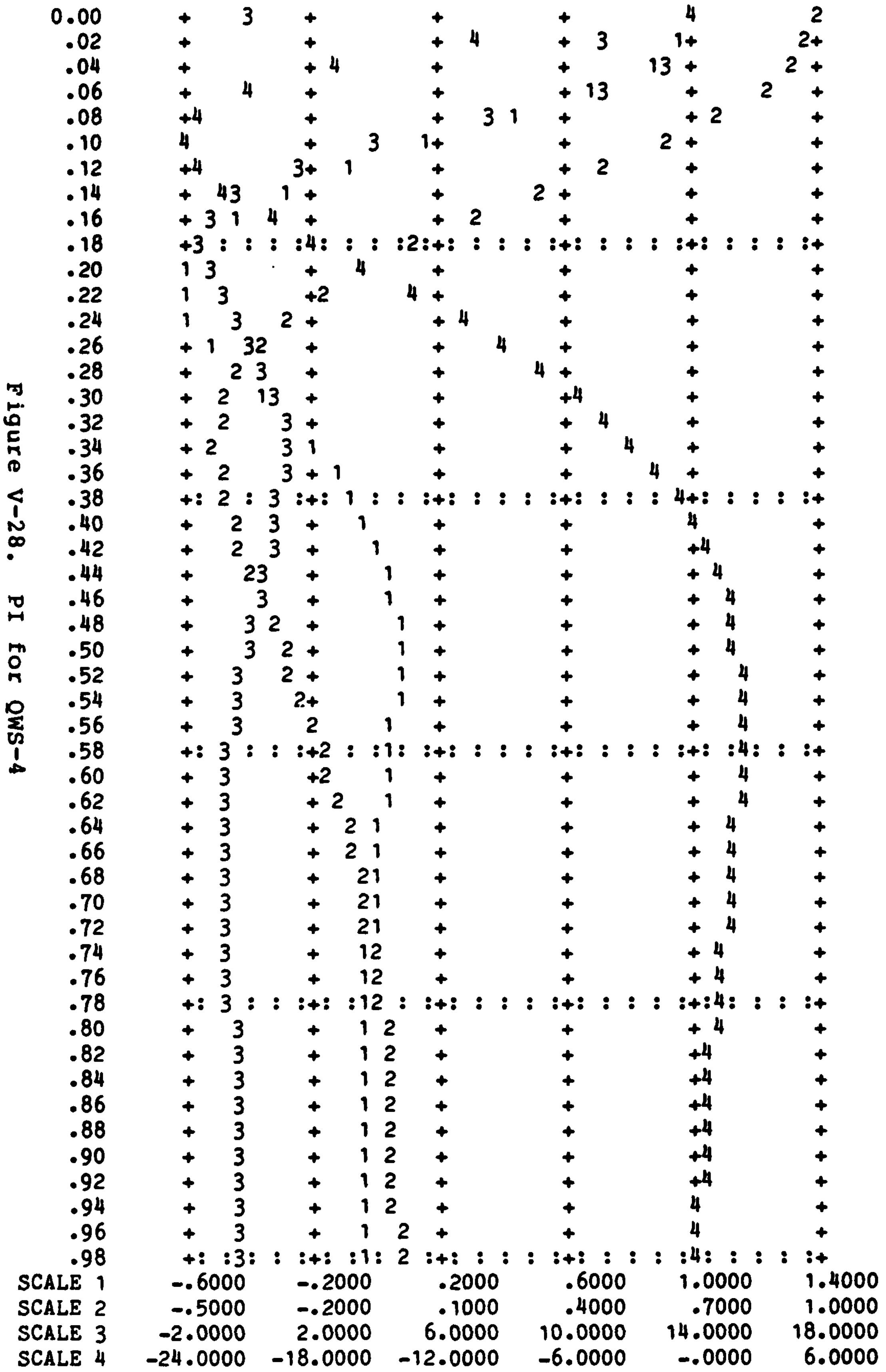


Figure V-28. PI for QWS-4

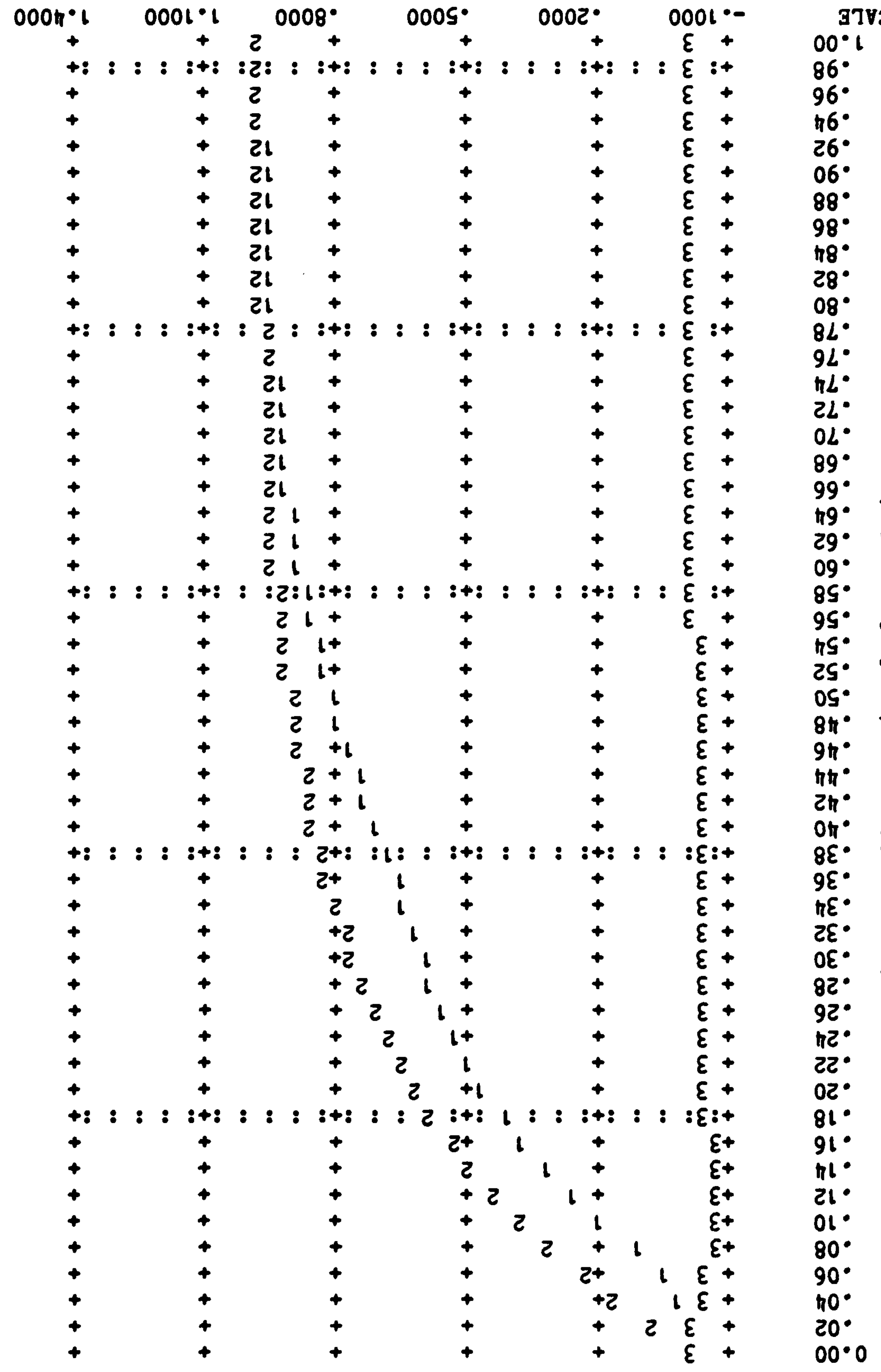


Figure V-29. CGT/PI-1 for QWS-4

```
    0.00      +  3      +         +         +         +         +
     .02      +  3  2   +         +         +         +         +
     .04      +  3     2+         +         +         +         +
     .06      + 13      +2        +         +         +         +
     .08      + 3       +   2     +         +         +         +
     .10      + 3       +     2   +         +         +         +
     .12      + 31      +       2 +         +         +         +
     .14      +3  1     +         2         +         +         +
     .16      +3    1   +         +2        +         +         +
     .18      3: : : : 1+: : : : :+: 2 : : :+: : : : :+: : : : :+
     .20      3         +1        +   2     +         +         +
     .22      3         +   1     +    2    +         +         +
     .24      3         +      1  +     2   +         +         +
     .26      3         +        1+      2  +         +         +
     .28      +3        +         + 1     2 +         +         +
     .30      +3        +         +   1    2+         +         +
     .32      +3        +         +     1  2+         +         +
     .34      +3        +         +       1 2         +         +
     .36      + 3       +         +        1+2        +         +
     .38      +:3: : : :+: : : : :+: : : : :+2 : : : :+: : : : :+
     .40      + 3       +         +         + 2       +         +
     .42      + 3       +         +         + 2       +         +
     .44      +  3      +         +         + 21      +         +
     .46      +  3      +         +         +  21     +         +
     .48      +  3      +         +         +  21     +         +
     .50      +  3      +         +         +  21     +         +
     .52      +  3      +         +         +   2     +         +
     .54      +  3      +         +         +   21    +         +
     .56      +  3      +         +         +   21    +         +
     .58      +: 3 : : :+: : : : :+: : : : :+: : :21 : :+: : : : :+
     .60      +  3      +         +         +    2    +         +
     .62      +  3      +         +         +    2    +         +
     .64      +  3      +         +         +    2    +         +
     .66      +  3      +         +         +    2    +         +
     .68      +  3      +         +         +    2    +         +
     .70      +  3      +         +         +    2    +         +
     .72      +  3      +         +         +    2    +         +
     .74      +  3      +         +         +    2    +         +
     .76      +  3      +         +         +    2    +         +
     .78      +: 3 : : :+: : : : :+: : : : :+: : 2 : :+: : : : :+
     .80      +  3      +         +         +    12   +         +
     .82      +  3      +         +         +     2   +         +
     .84      +  3      +         +         +     2   +         +
     .86      +  3      +         +         +     2   +         +
     .88      +  3      +         +         +     2   +         +
     .90      +  3      +         +         +     2   +         +
     .92      +  3      +         +         +     2   +         +
     .94      +  3      +         +         +     2   +         +
     .96      +  3      +         +         +     2   +         +
     .98      +: 3 : : :+: : : : :+: : : : :+: : :2: :+: : : : :+
    1.00      +  3      +         +         +     2   +         +
SCALE        -.1000     .2000     .5000     .8000    1.1000    1.4000
```

Figure V-30. CGT/PI-2 for QWS-4

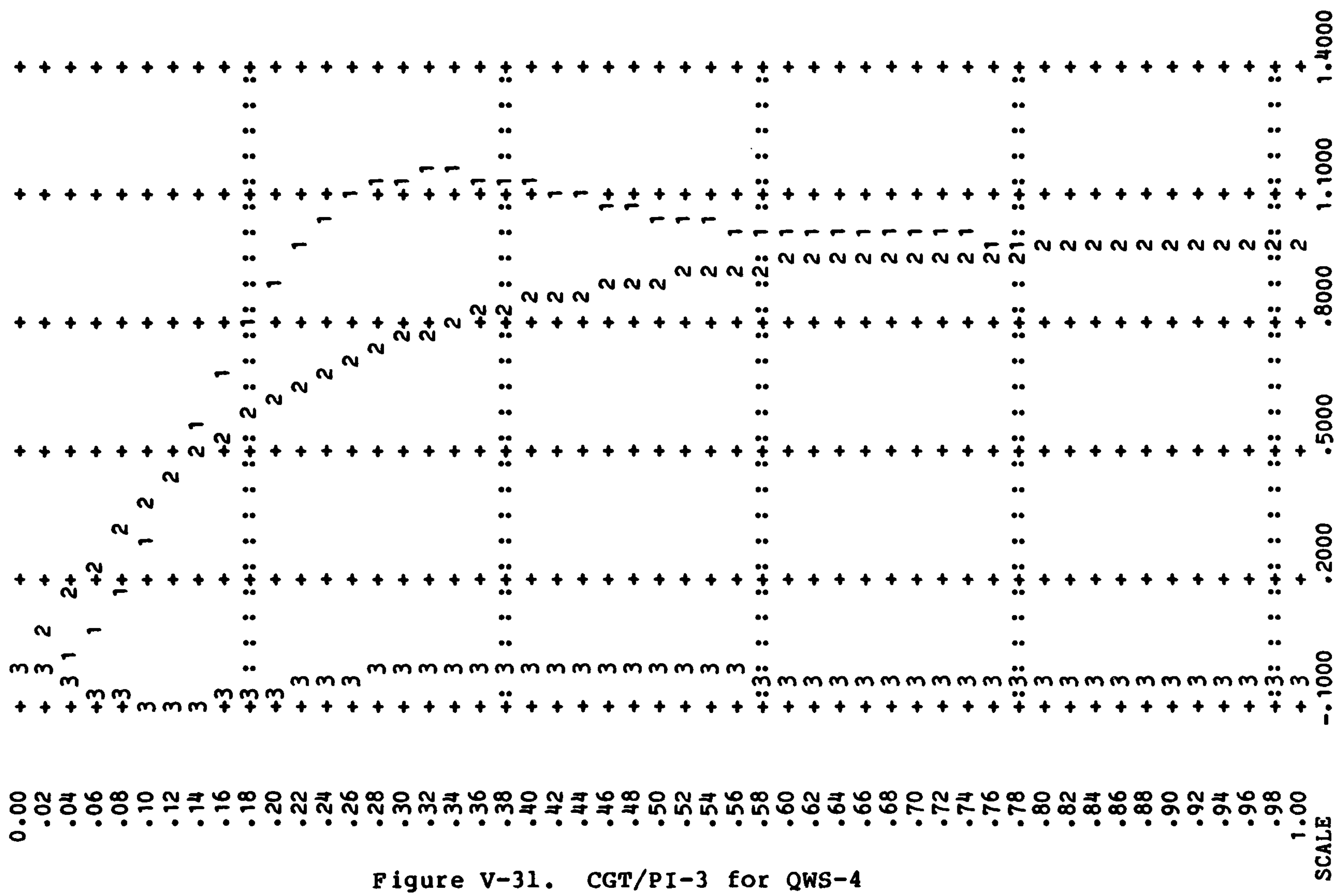


Figure V-31. CGT/PI-3 for QWS-4

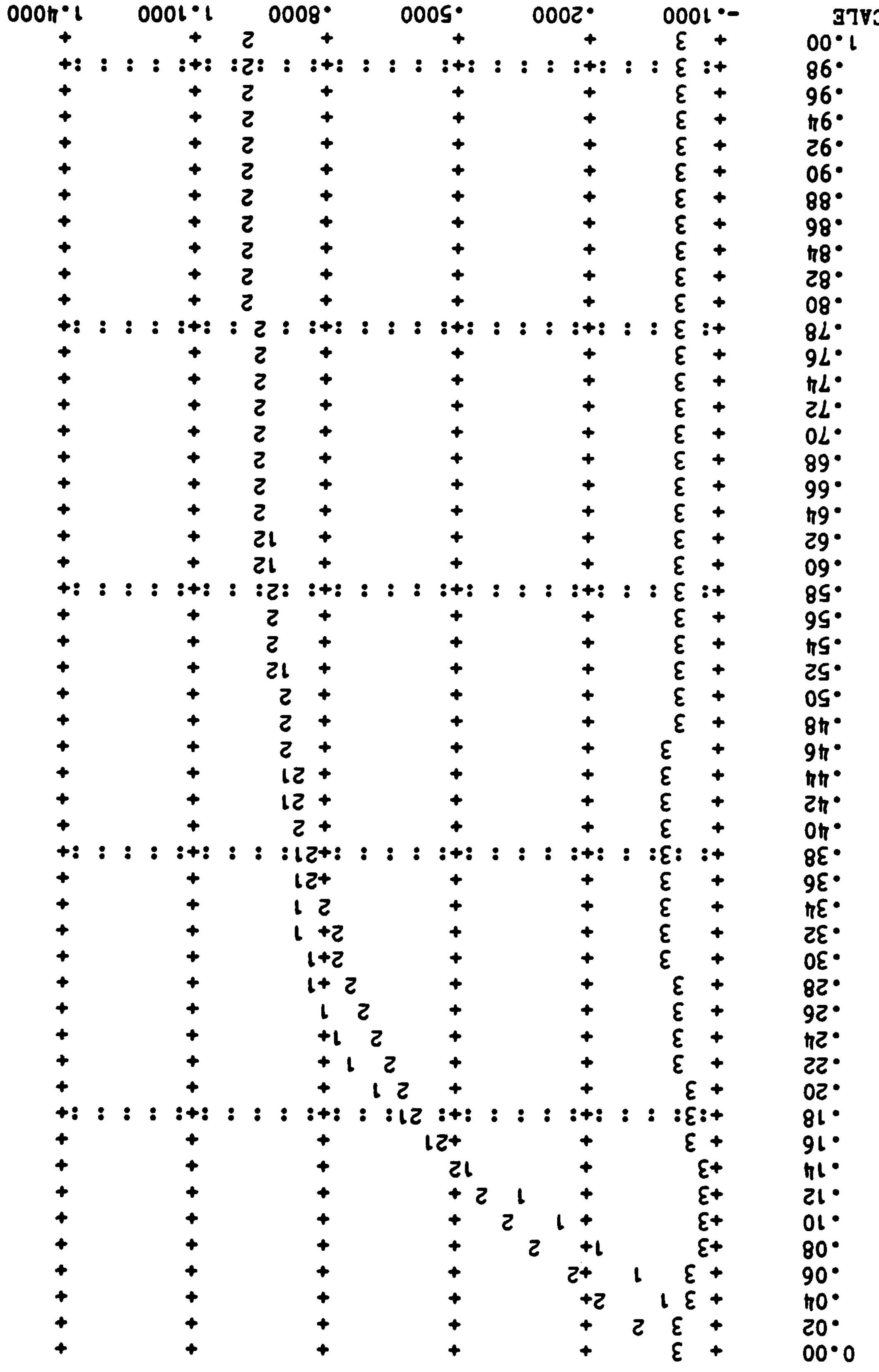


Figure V-32. CGT/PI-4 for QWS-4

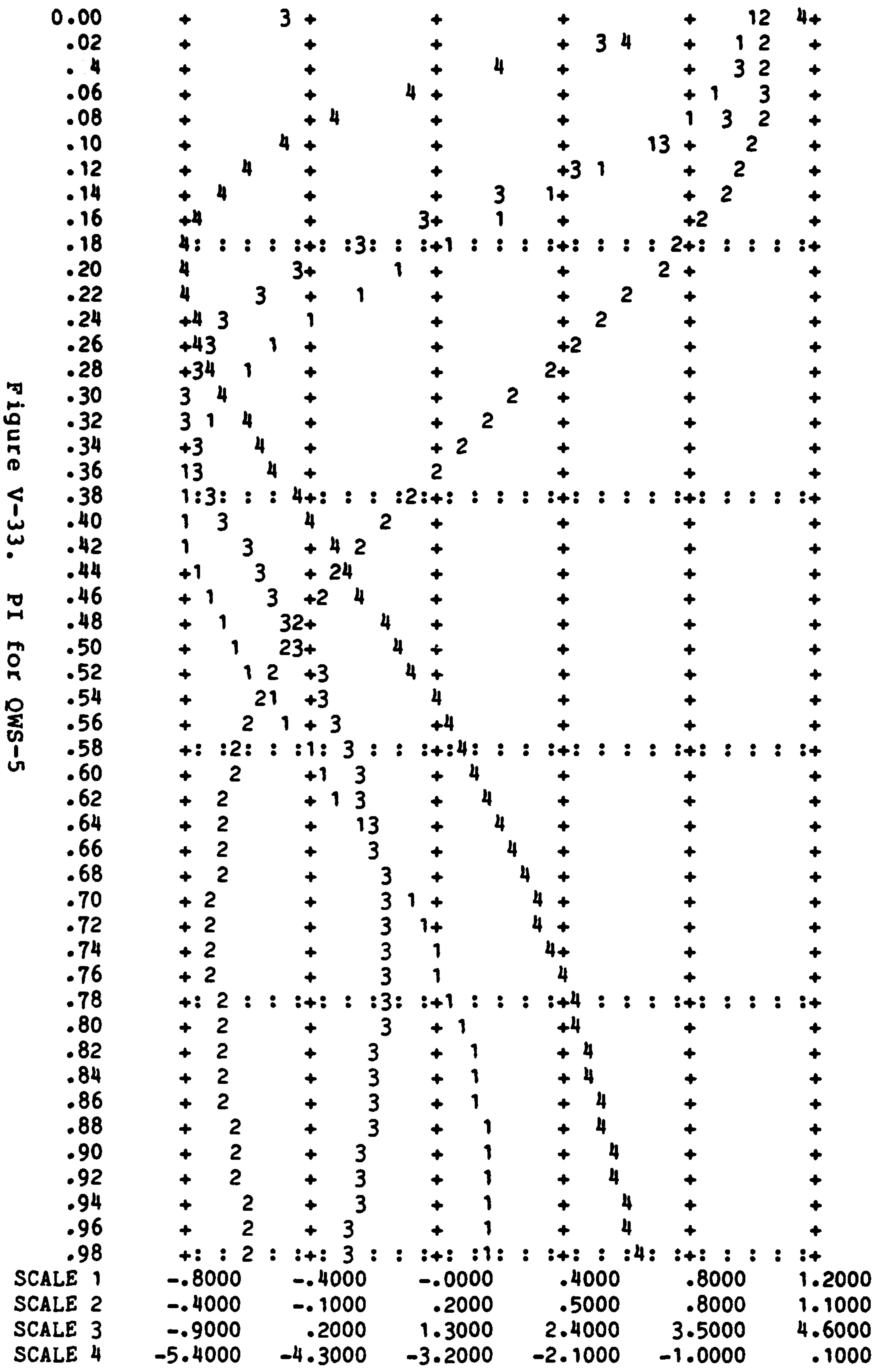


Figure V-33. PI for QWS-5

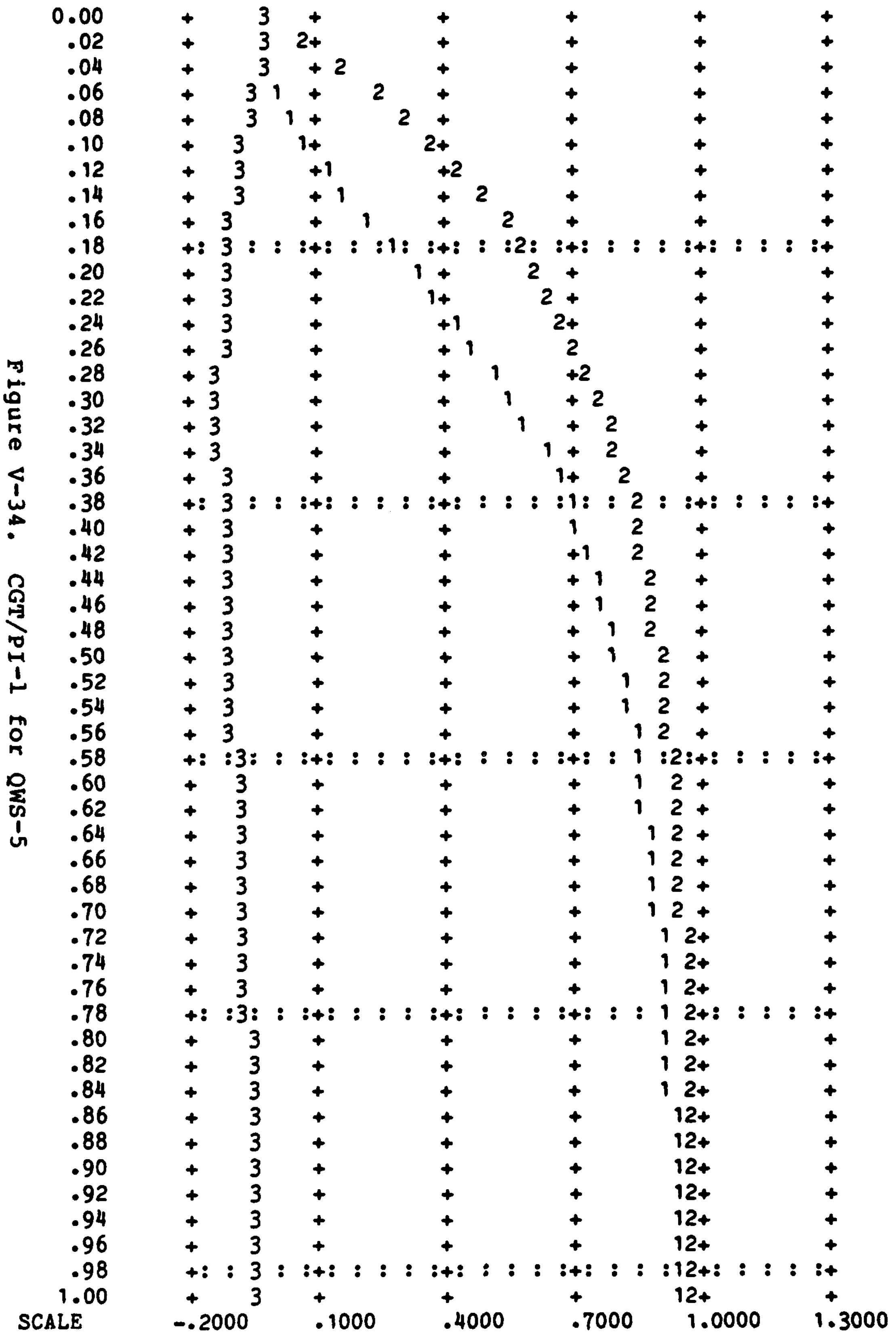


Figure V-34. CGT/PI-1 for QWS-5

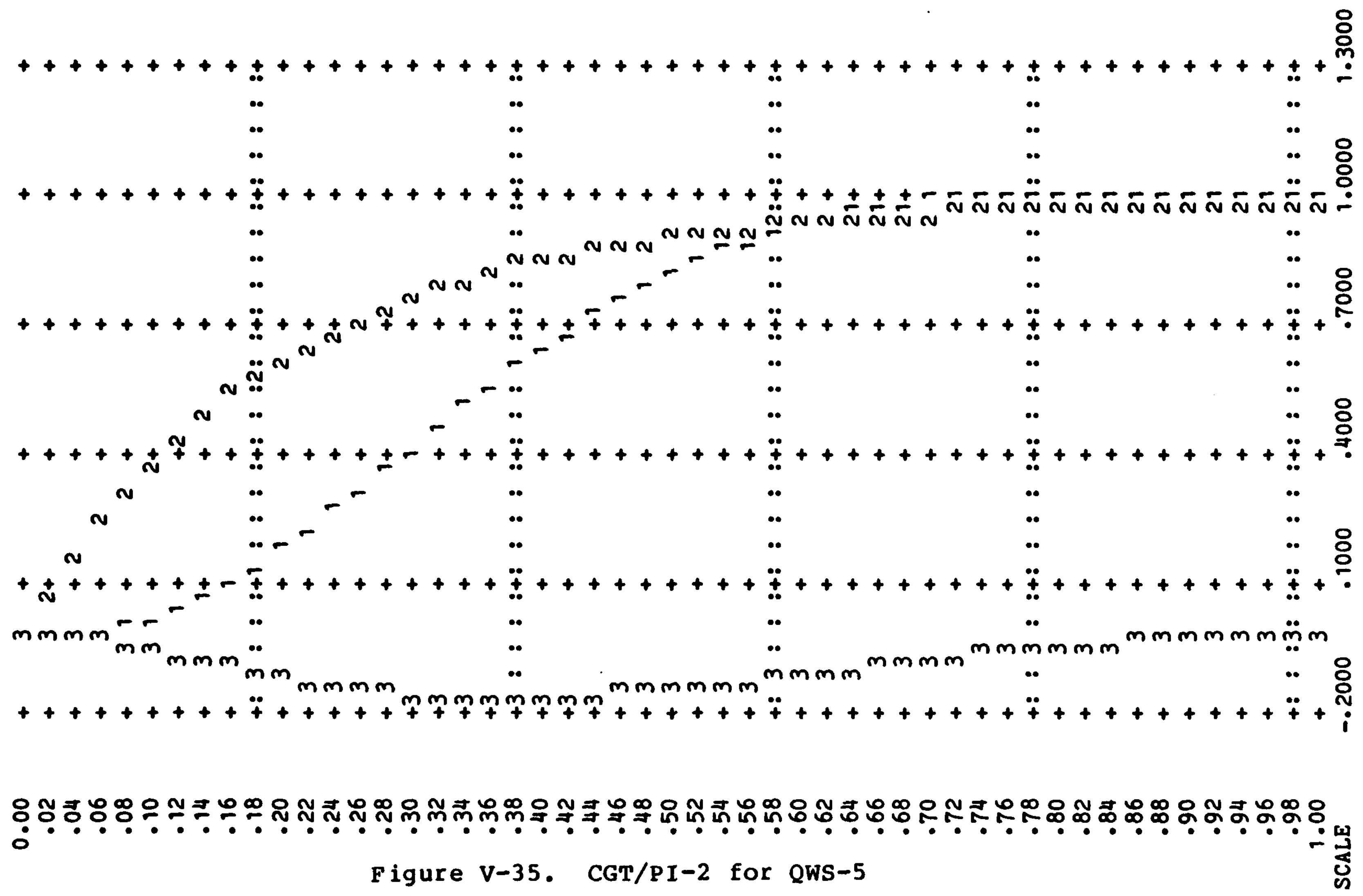


Figure V-35. CGT/PI-2 for QWS-5

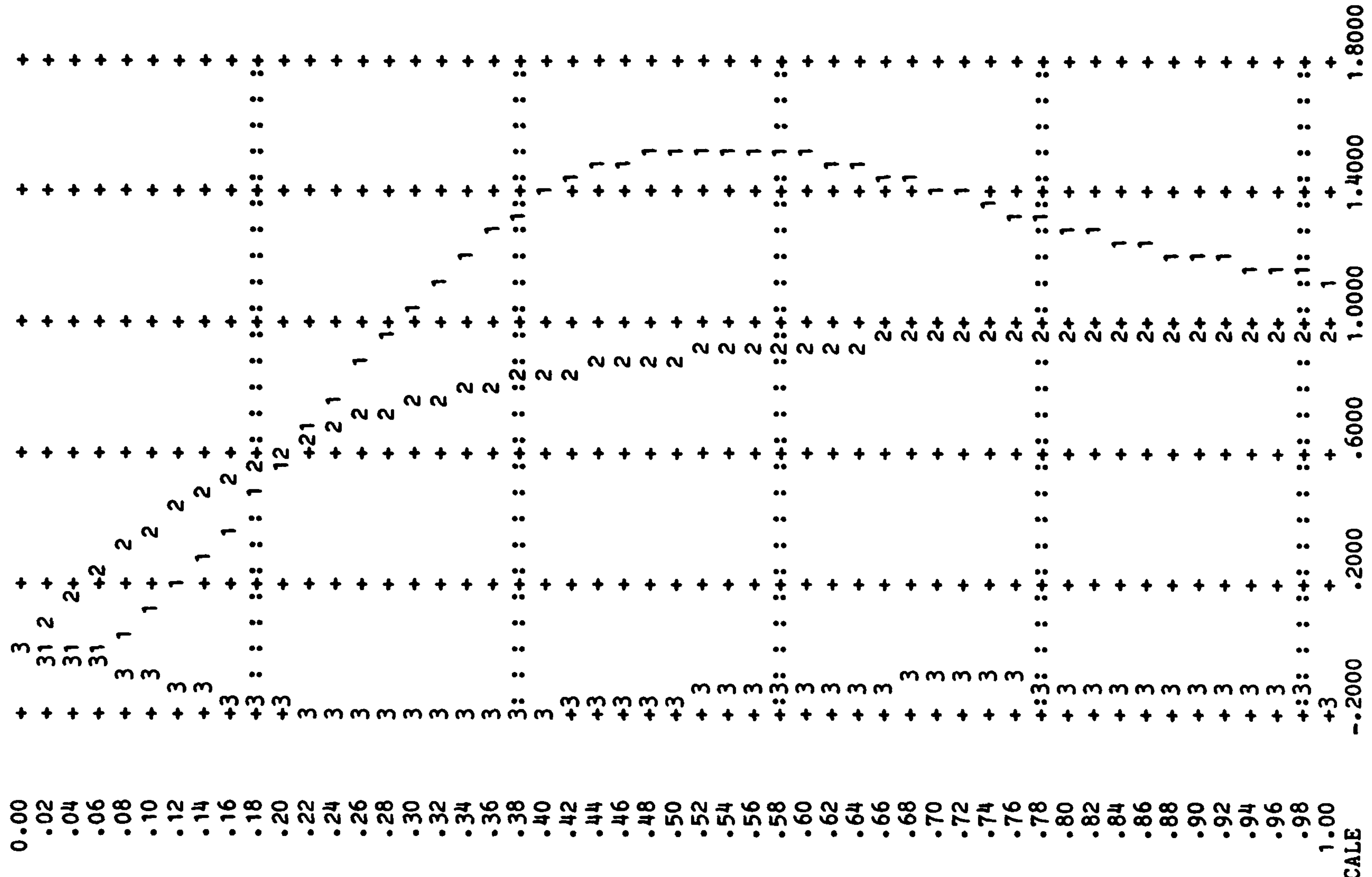


Figure V-36. CGT/PI-3 for QWS-5

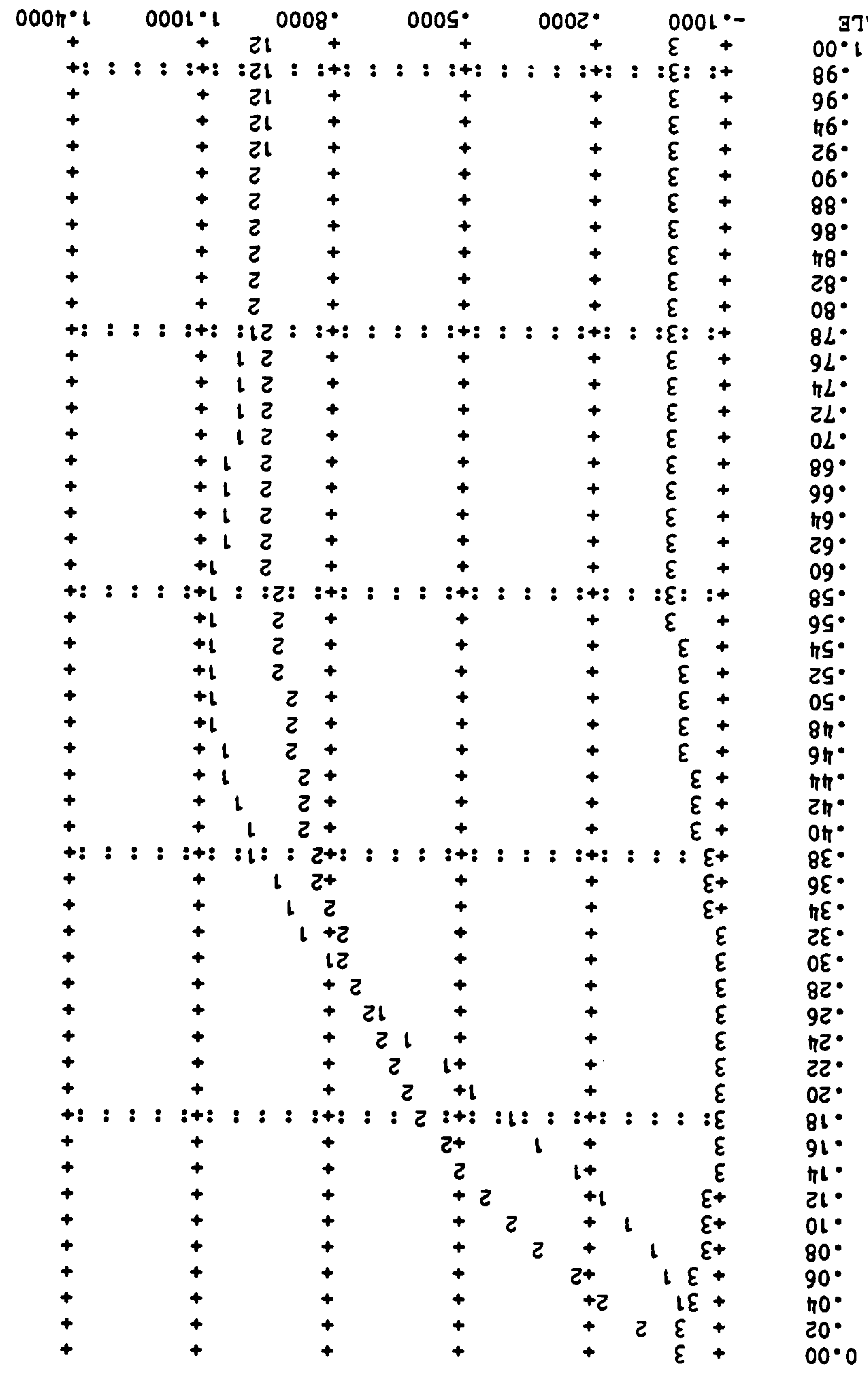


Figure V-37. CGT/PI-4 for QWS-5

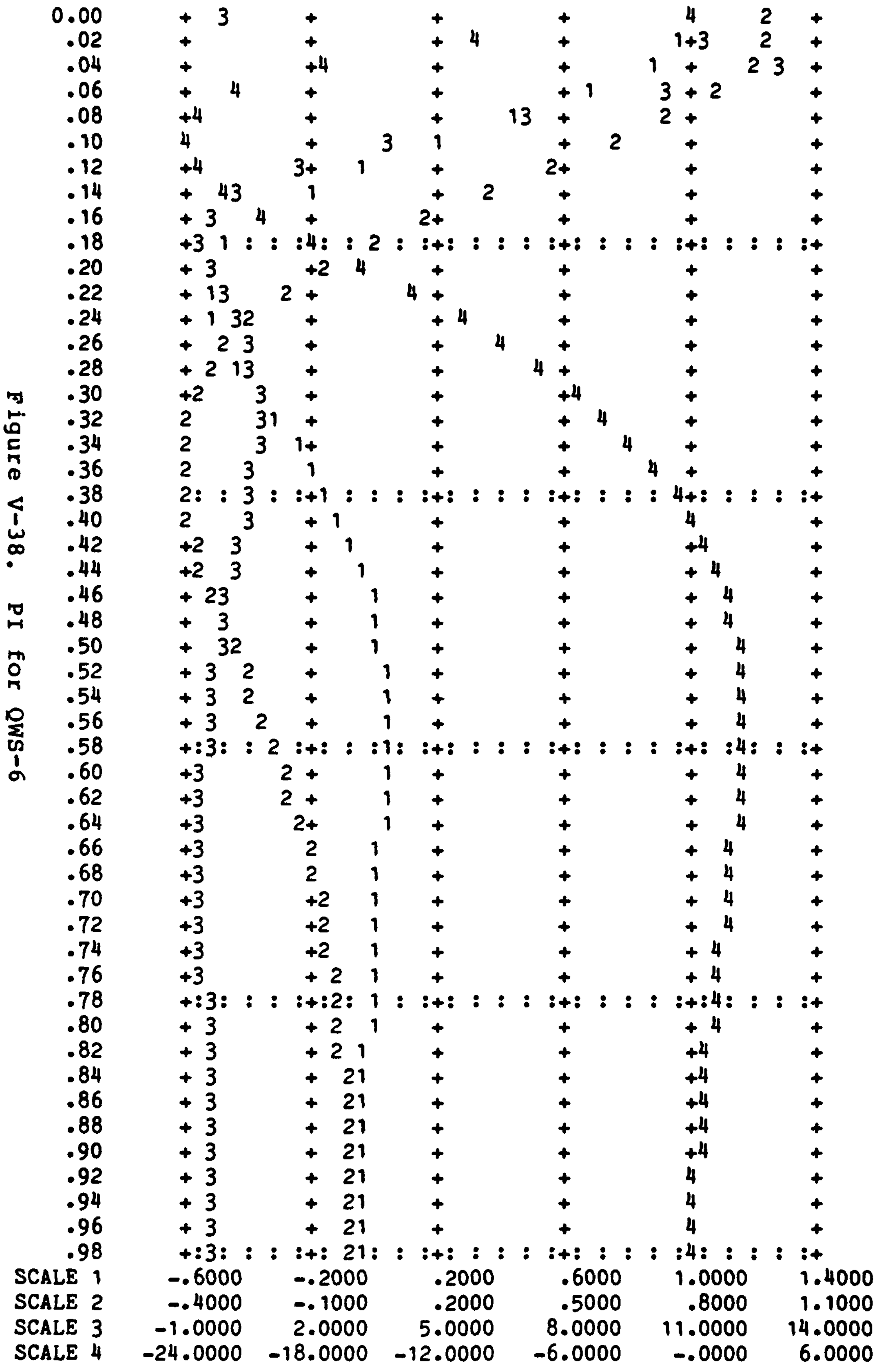


Figure V-38. PI for QWS-6

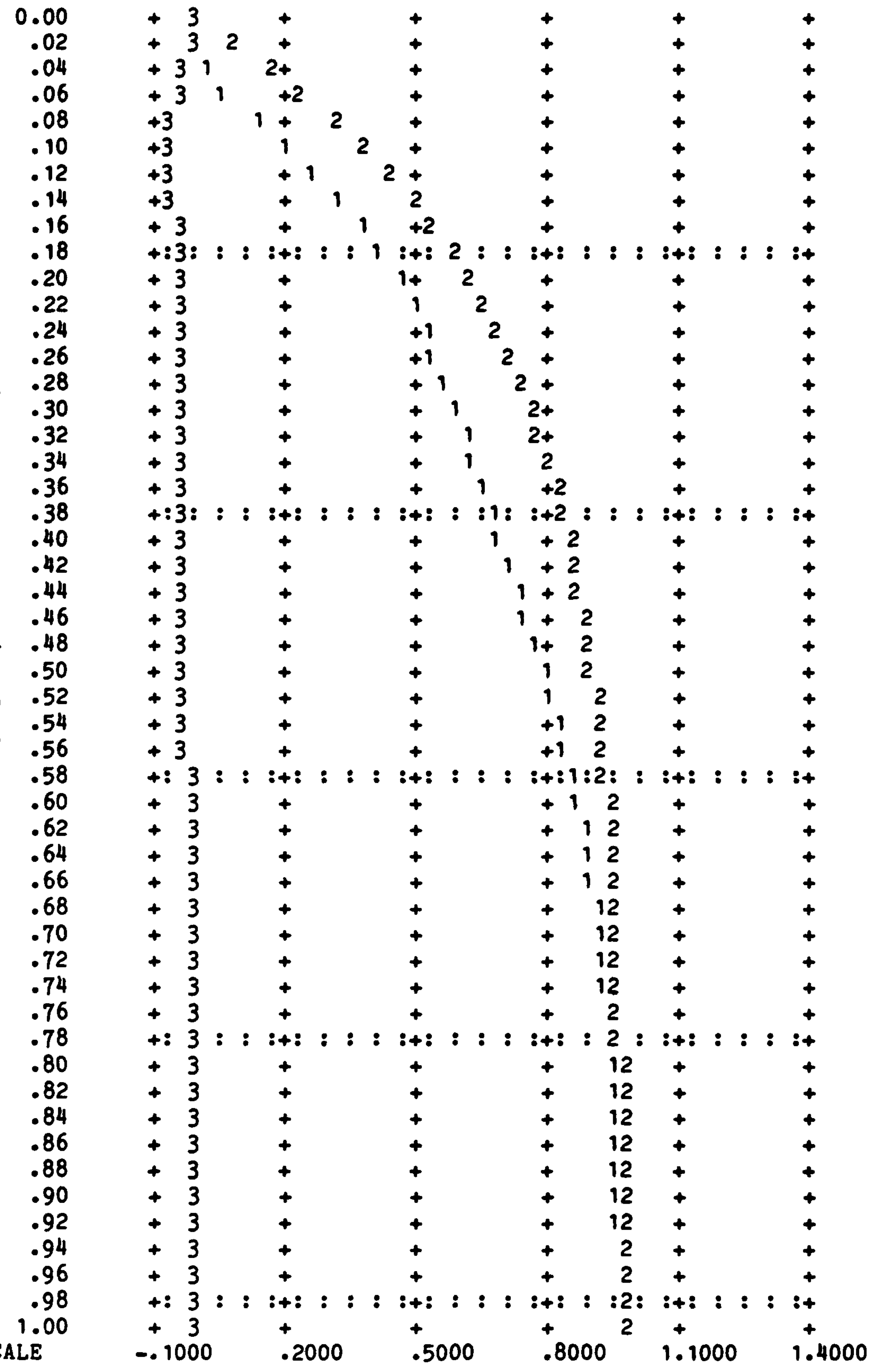


Figure V-39. CGT/PI-1 for QWS-6

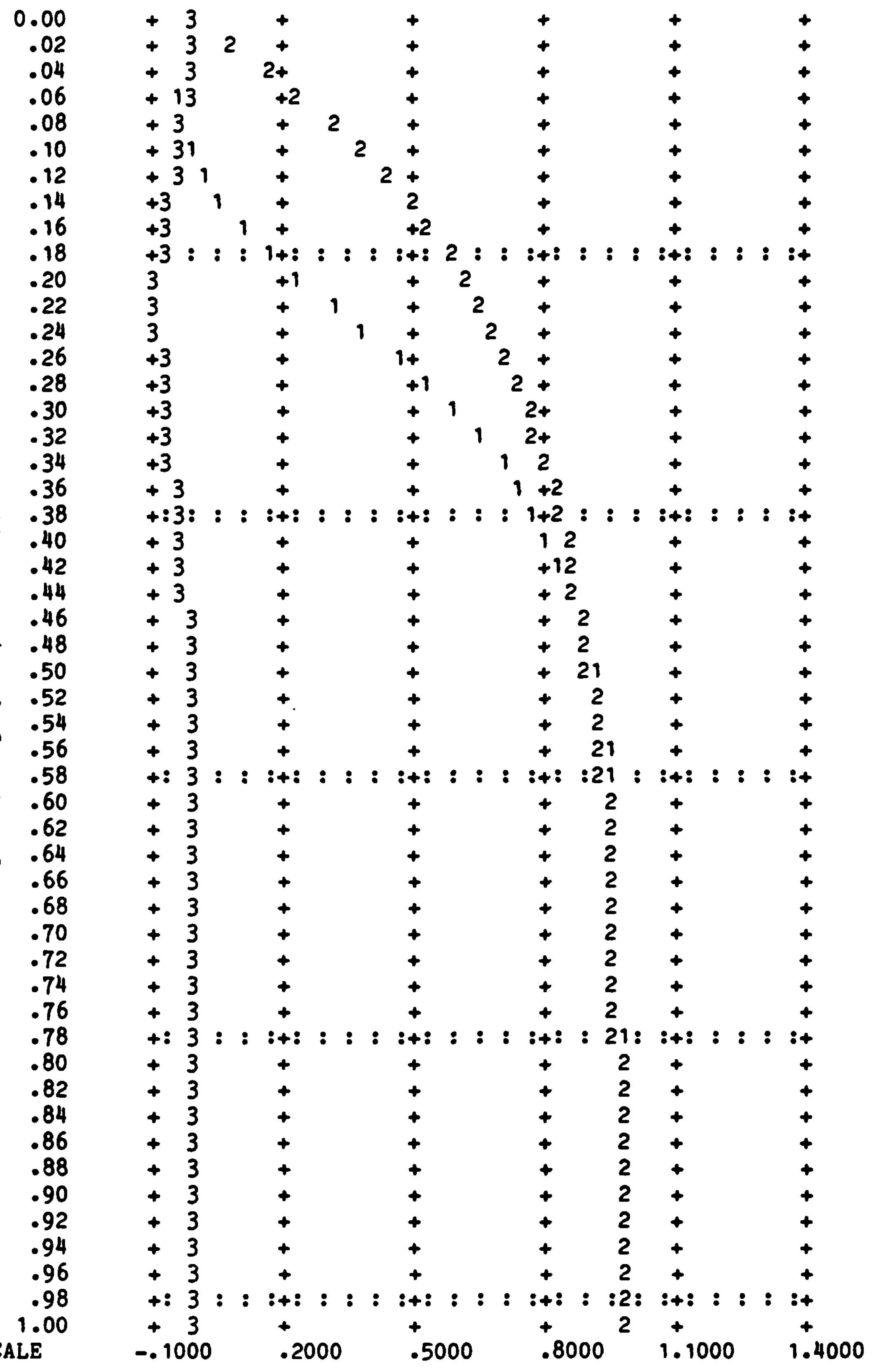


Figure V-40. CGT/PI-2 for QWS-6

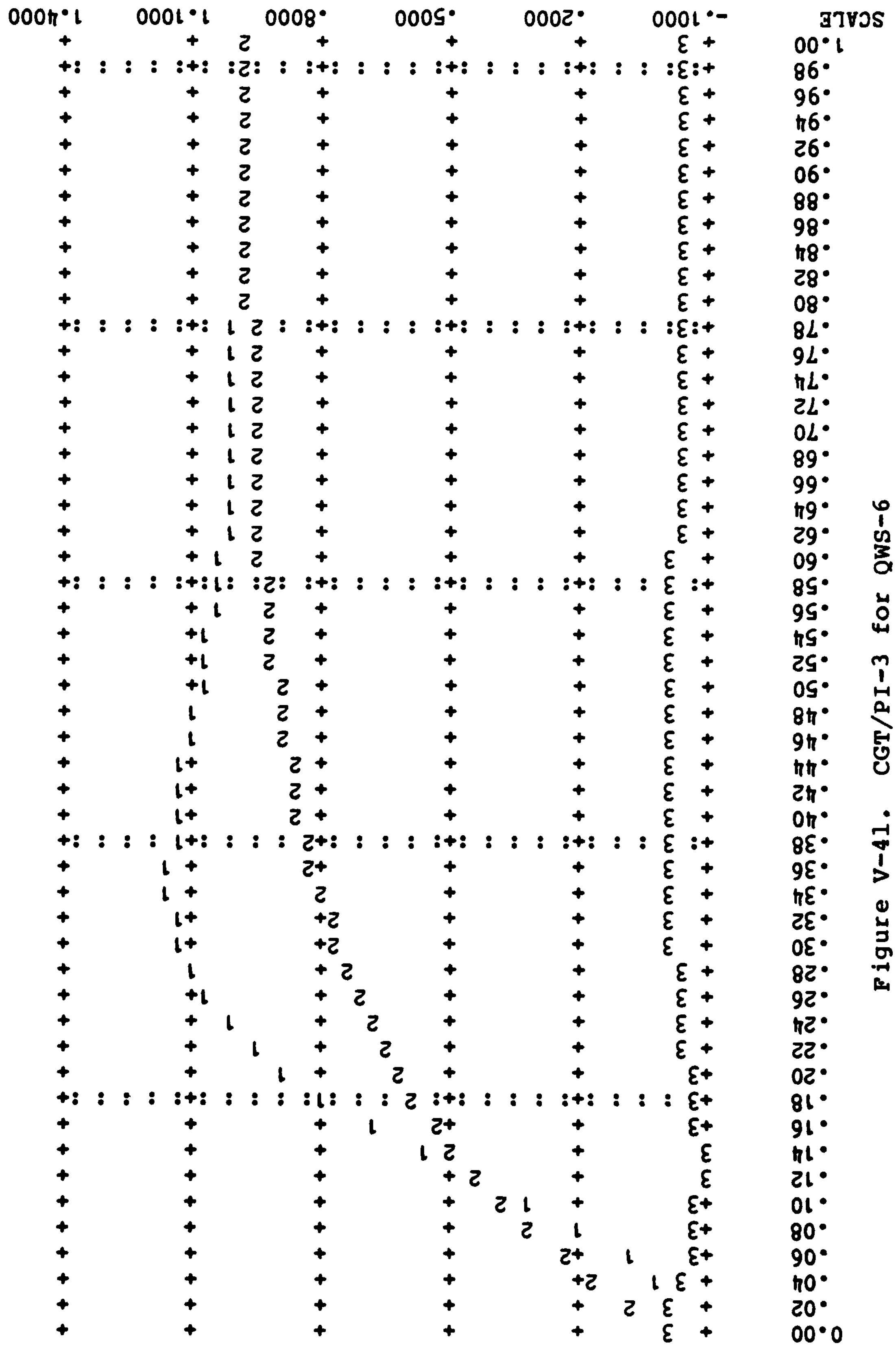


Figure V-41. CGT/PI-3 for QWS-6

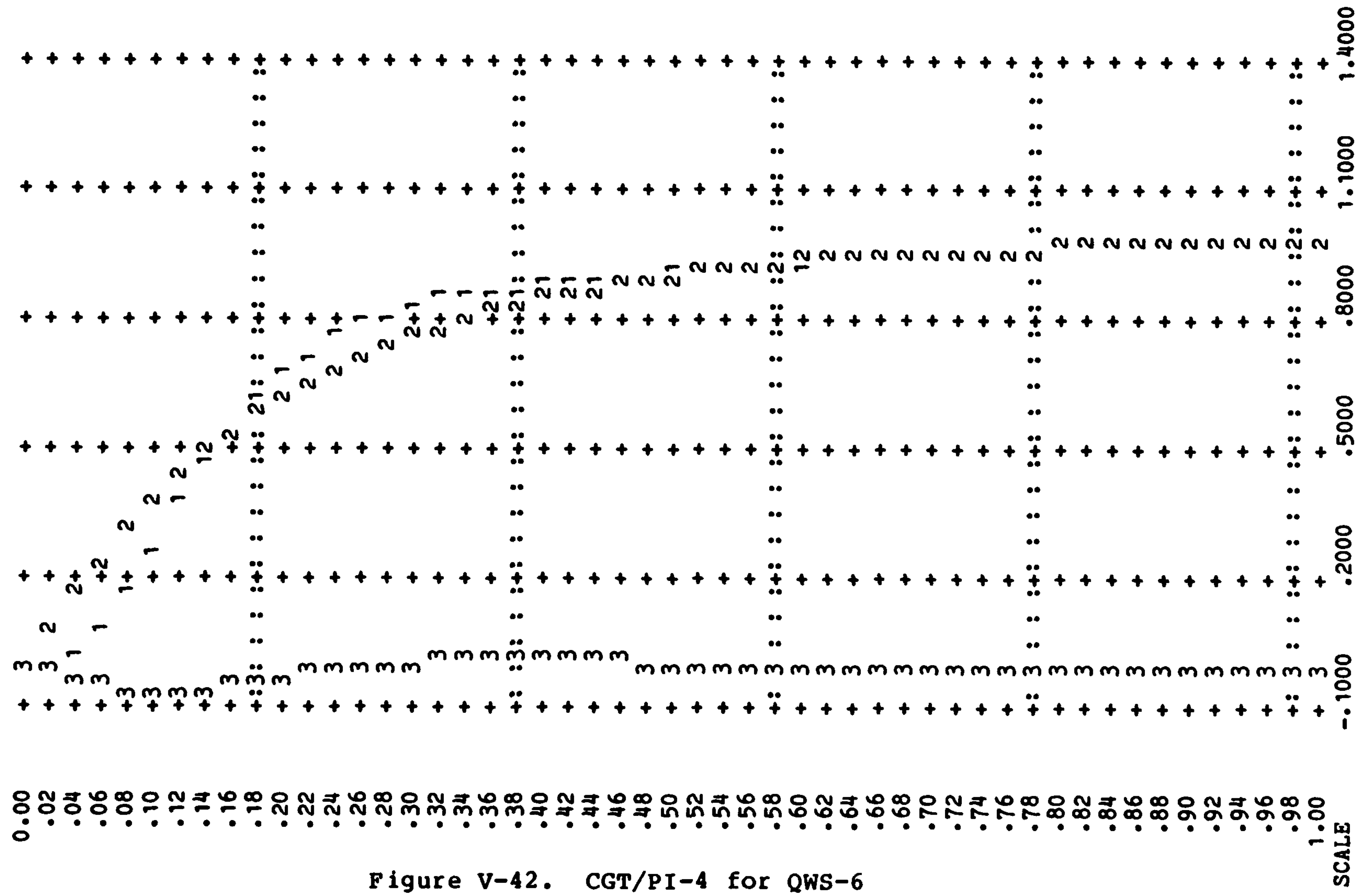


Figure V-42. CGT/PI-4 for QWS-6

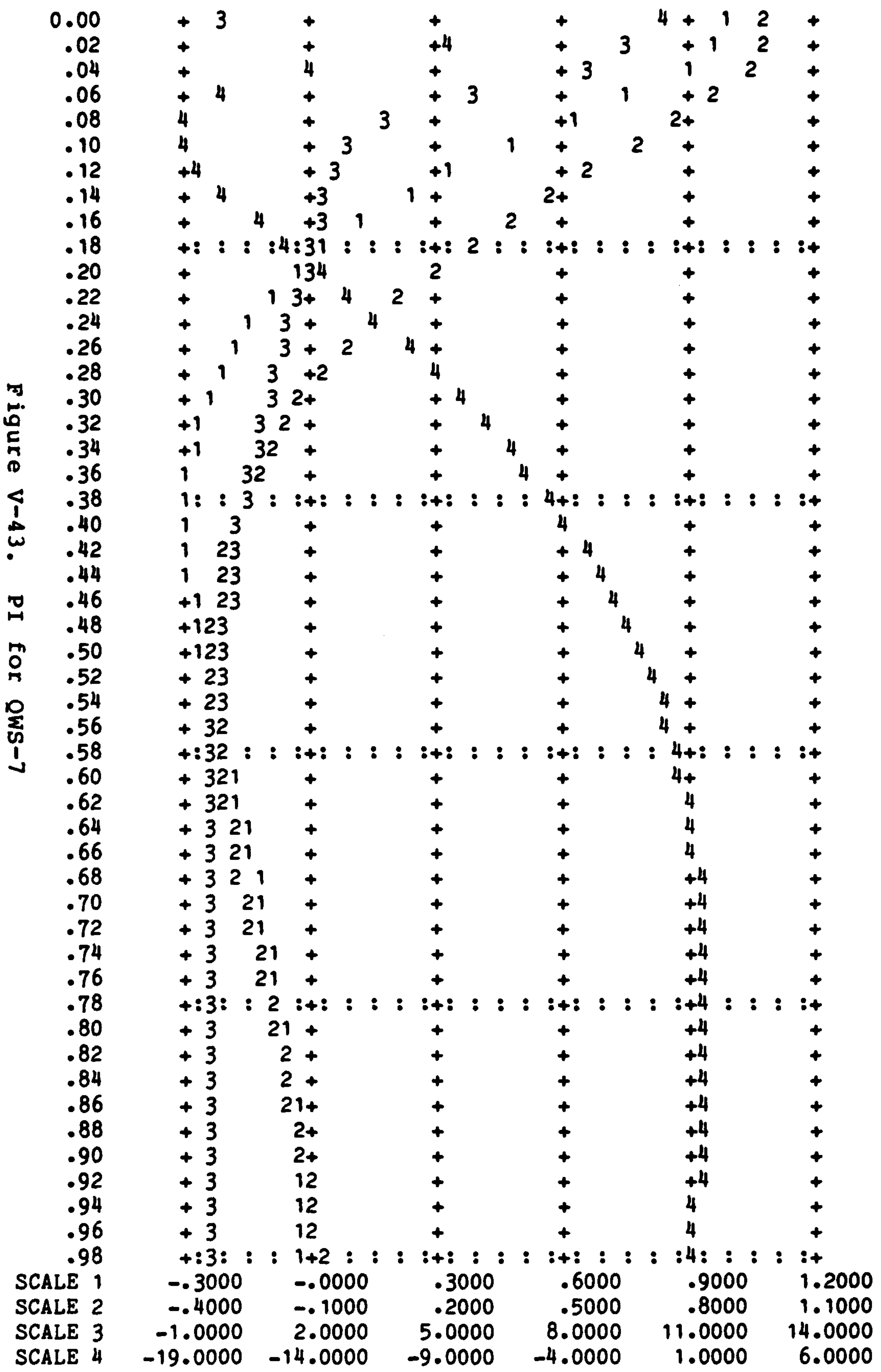


Figure V-43. PI for QWS-7

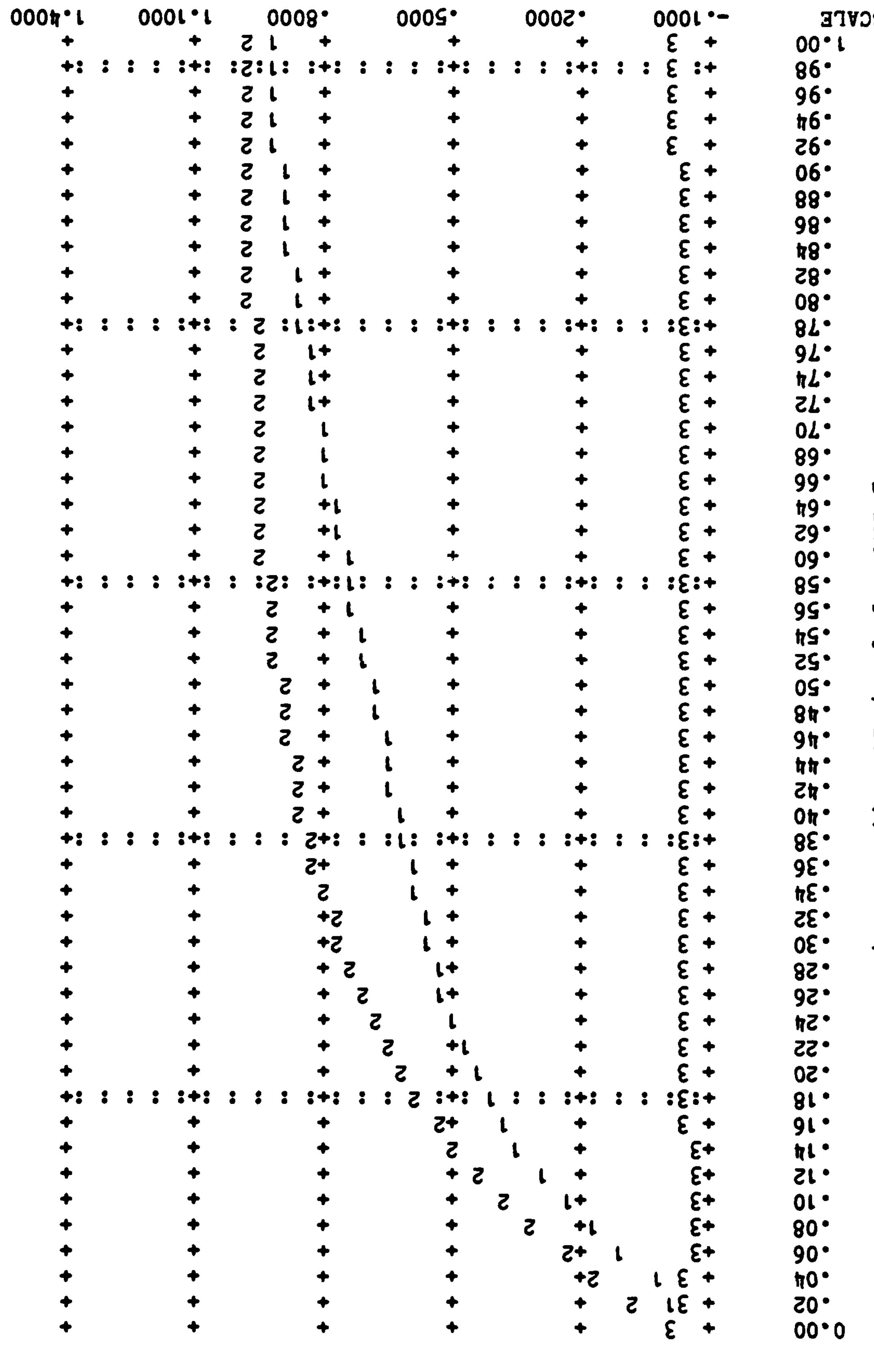


Figure V-44. CGT/PI-1 for QWS-7

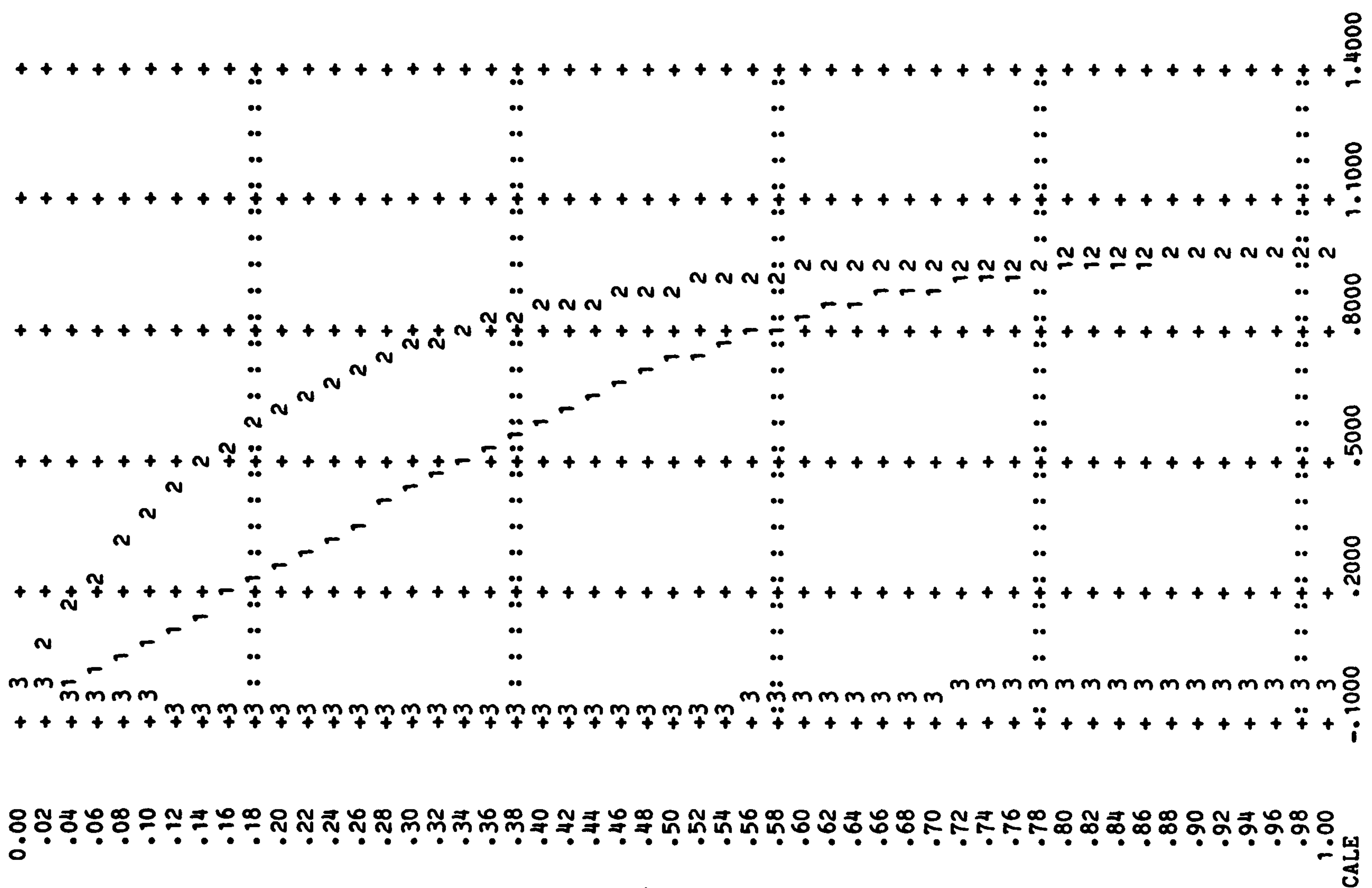


Figure V-45. CGT/PI-2 for QWS-7

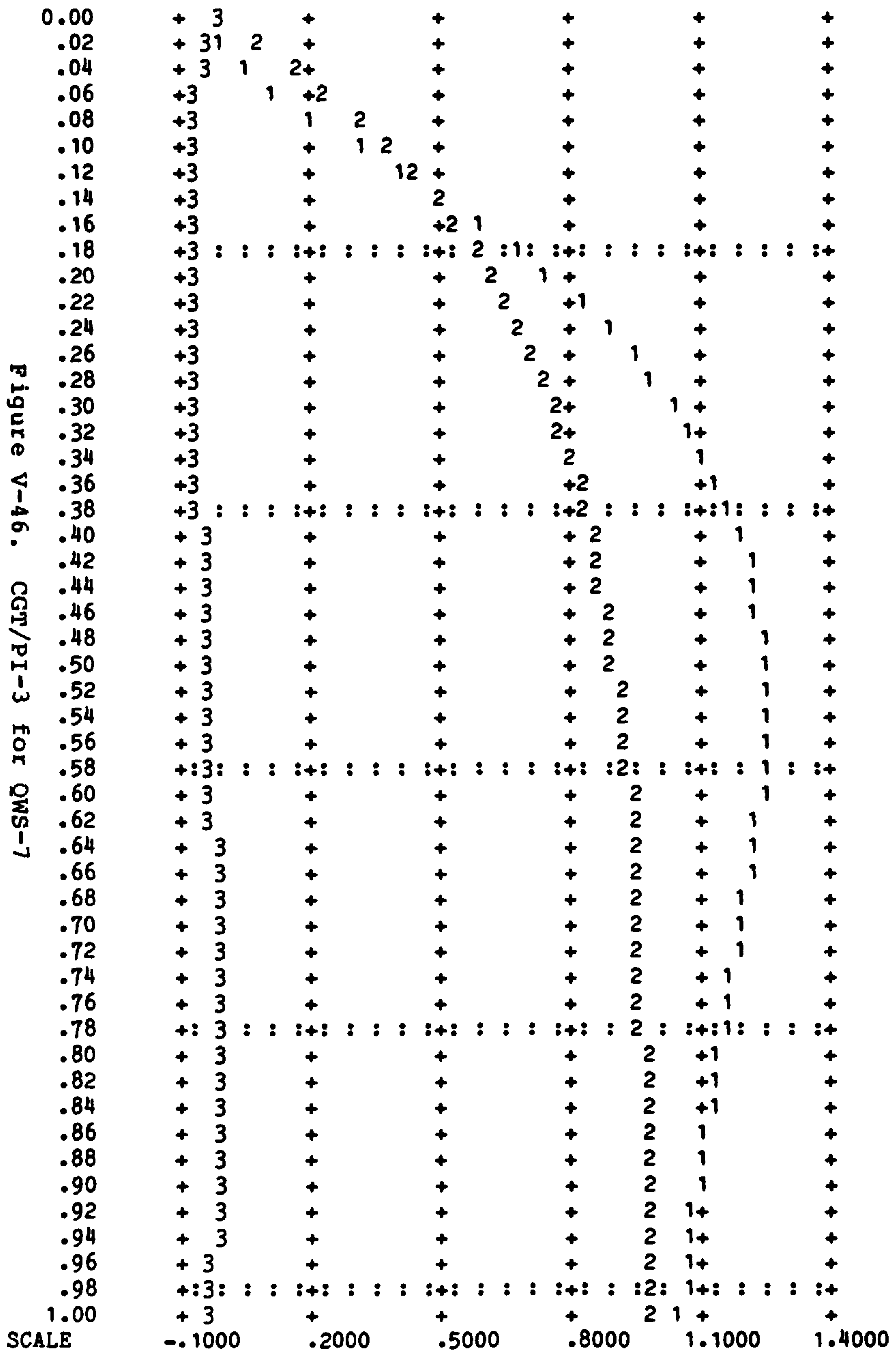


Figure V-46. CGT/PI-3 for QWS-7

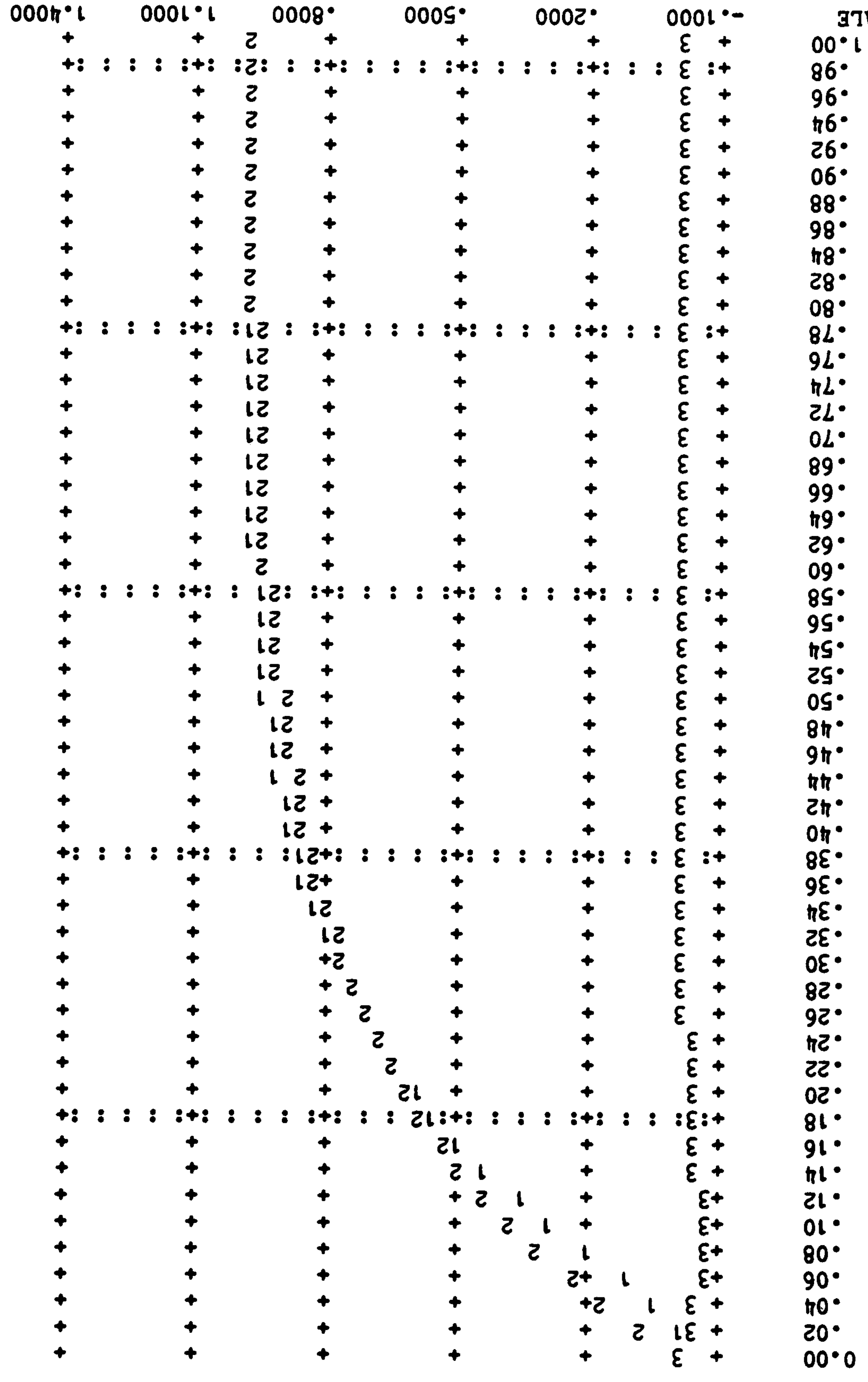


Figure V-47. CGT/PI-4 for QWS-7

```
   0.00      +  3      +         +         +      4  1     2   +
    .02      +         +         +   4     +        3+     2   +
    .04      +         +   4     +         +       1 +     2  3+
    .06      +       4 +         +         +    1    +   23    +
    .08      +   4     +         +         1     3   + 2       +
    .10      + 4       +         +    3    +         2         +
    .12      +4        +     3   1         +      2  +         +
    .14      +4       3+    1    +         +   2     +         +
    .16      + 4 3     +1        +         2         +         +
    .18      +3 :4: 1 :+: : : : :+: : : 2 :+: : : : :+: : : : :+
    .20      3   1 4   +         +  2      +         +         +
    .22      3 1     4 +         2         +         +         +
    .24      31        +4     2  +         +         +         +
    .26      1 3       +  42     +         +         +         +
    .28      1  3      + 2  4    +         +         +         +
    .30      +1  3     2      4  +         +         +         +
    .32      + 1  3  2 +        4+         +         +         +
    .34      +  1  23  +         +4        +         +         +
    .36      +   12 3  +         +  4      +         +         +
    .38      +: :2:1:3:+: : : : :+: :4: : :+: : : : :+: : : : :+
    .40      +  2   13 +         +     4   +         +         +
    .42      + 2     13+         +      4  +         +         +
    .44      + 2      31         +        4+         +         +
    .46      + 2     3 +1        +         4         +         +
    .48      +2      3 + 1       +         +4        +         +
    .50      +2      3 +  1      +         + 4       +         +
    .52      + 2     3 +   1     +         +  4      +         +
    .54      + 2    3  +    1    +         +  4      +         +
    .56      + 2    3  +    1    +         +   4     +         +
    .58      +:2: :3: :+: : :1: :+: : : : :+: : 4 : :+: : : : :+
    .60      +  2  3   +     1   +         +    4    +         +
    .62      +  2 3    +     1   +         +     4   +         +
    .64      +  2 3    +      1  +         +     4   +         +
    .66      +   23    +      1  +         +      4  +         +
    .68      +   3     +      1  +         +      4  +         +
    .70      +   32    +      1  +         +      4  +         +
    .72      +  3 2    +      1  +         +       4 +         +
    .74      +  3  2   +      1  +         +       4 +         +
    .76      +  3  2   +      1  +         +       4 +         +
    .78      +: 3 :2: :+: : :1: :+: : : : :+: : : :4:+: : : : :+
    .80      + 3    2  +     1   +         +       4 +         +
    .82      + 3    2  +     1   +         +        4+         +
    .84      + 3     2 +     1   +         +        4+         +
    .86      + 3     2 +     1   +         +        4+         +
    .88      + 3     2 +     1   +         +        4+         +
    .90      + 3      2+     1   +         +        4+         +
    .92      + 3      2+     1   +         +        4+         +
    .94      + 3      2+    1    +         +        4+         +
    .96      + 3       2    1    +         +        4+         +
    .98      +:3: : : :2: : 1 : :+: : : : :+: : : : 4+: : : : :+
SCALE 1    -.6000    -.2000     .2000     .6000    1.0000    1.4000
SCALE 2    -.4000    -.1000     .2000     .5000     .8000    1.1000
SCALE 3    -.6000    1.3000    3.2000    5.1000    7.0000    8.9000
SCALE 4  -15.0000  -11.0000   -7.0000   -3.0000    1.0000    5.0000
```

Figure V-48. PI for QWS-8

```
    0.00    +  3      +         +         +         +         +
     .02    +  3  2   +         +         +         +         +
     .04    + 3 1    2+         +         +         +         +
     .06    + 3  1    +2        +         +         +         +
     .08    +3    1   +   2     +         +         +         +
     .10    +3      1 +     2   +         +         +         +
     .12    +3        +1      2 +         +         +         +
     .14    +3        +  1      2         +         +         +
     .16    +3        +    1    +2        +         +         +
     .18    +3 : : : :+: : : 1 :+: 2 : : :+: : : : :+: : : : :+
     .20    +3        +       1 +   2     +         +         +
     .22    +3        +         1    2    +         +         +
     .24    +3        +         +1    2   +         +         +
     .26    +3        +         + 1    2  +         +         +
     .28    +3        +         +  1    2 +         +         +
     .30    +3        +         +  1     2+         +         +
     .32    +3        +         +   1    2+         +         +
     .34    + 3       +         +    1    2         +         +
     .36    + 3       +         +    1    +2        +         +
     .38    +:3: : : :+: : : : :+: : :1: :+2 : : : :+: : : : :+
     .40    + 3       +         +     1   + 2       +         +
     .42    + 3       +         +      1  + 2       +         +
     .44    + 3       +         +      1  + 2       +         +
     .46    + 3       +         +       1 +  2      +         +
     .48    + 3       +         +       1 +  2      +         +
     .50    + 3       +         +        1+  2      +         +
     .52    + 3       +         +        1+   2     +         +
     .54    + 3       +         +         1   2     +         +
     .56    + 3       +         +         1   2     +         +
     .58    +:3: : : :+: : : : :+: : : : :+1 :2: : :+: : : : :+
     .60    + 3       +         +         +1   2    +         +
     .62    + 3       +         +         + 1  2    +         +
     .64    + 3       +         +         + 1  2    +         +
     .66    + 3       +         +         + 1  2    +         +
     .68    + 3       +         +         +  1 2    +         +
     .70    + 3       +         +         +  1 2    +         +
     .72    + 3       +         +         +  1 2    +         +
     .74    + 3       +         +         +  1 2    +         +
     .76    + 3       +         +         +   12    +         +
     .78    +: 3 : : :+: : : : :+: : : : :+: :12 : :+: : : : :+
     .80    +  3      +         +         +   1 2   +         +
     .82    +  3      +         +         +   1 2   +         +
     .84    +  3      +         +         +    12   +         +
     .86    +  3      +         +         +    12   +         +
     .88    +  3      +         +         +    12   +         +
     .90    +  3      +         +         +    12   +         +
     .92    +  3      +         +         +    12   +         +
     .94    +  3      +         +         +    12   +         +
     .96    +  3      +         +         +    12   +         +
     .98    +: 3 : : :+: : : : :+: : : : :+: : 12: :+: : : : :+
    1.00    +  3      +         +         +    12   +         +
SCALE      -.1000     .2000     .5000     .8000    1.1000    1.4000
```

Figure V-49. CGT/PI-1 for QWS-8

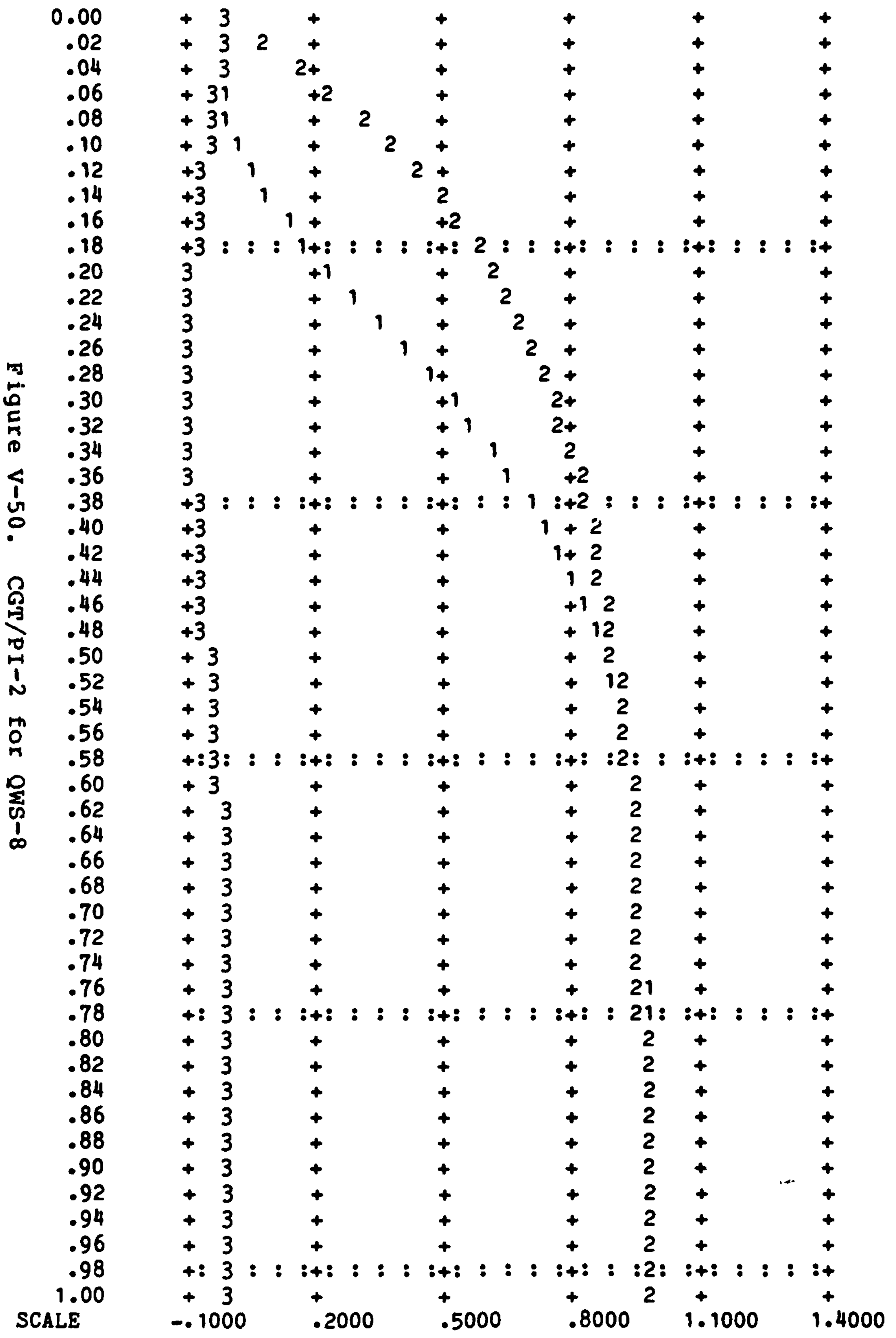


Figure V-50. CGT/PI-2 for QWS-8

```
 0.00   +  3      +         +         +         +         +
  .02   +  3  2   +         +         +         +         +
  .04   + 3 1    2+         +         +         +         +
  .06   + 3  1    +2        +         +         +         +
  .08   +3      1 +   2     +         +         +         +
  .10   3         +1    2   +         +         +         +
  .12   3         +    1  2 +         +         +         +
  .14   3         +        12         +         +         +
  .16   3         +         +2 1      +         +         +
  .18   3: : : : :+: : : : :+: 2 : 1 :+: : : : :+: : : : :+
  .20   3         +         +   2     1         +         +
  .22   3         +         +    2    +   1     +         +
  .24   3         +         +     2   +      1  +         +
  .26   3         +         +      2  +        1+         +
  .28   +3        +         +       2 +         +1        +
  .30   +3        +         +        2+         +  1      +
  .32   +3        +         +        2+         +   1     +
  .34   +3        +         +         2         +    1    +
  .36   + 3       +         +         +2        +     1   +
  .38   +:3: : : :+: : : : :+: : : : :+2 : : : :+: : :1: :+
  .40   + 3       +         +         + 2       +     1   +
  .42   + 3       +         +         + 2       +     1   +
  .44   + 3       +         +         + 2       +     1   +
  .46   + 3       +         +         +  2      +    1    +
  .48   + 3       +         +         +  2      +    1    +
  .50   + 3       +         +         +  2      +    1    +
  .52   + 3       +         +         +   2     +   1     +
  .54   + 3       +         +         +   2     +   1     +
  .56   + 3       +         +         +   2     +  1      +
  .58   +:3: : : :+: : : : :+: : : : :+: : 2 : :+:1: : : :+
  .60   + 3       +         +         +    2    + 1       +
  .62   + 3       +         +         +    2    +1        +
  .64   + 3       +         +         +    2    +1        +
  .66   + 3       +         +         +    2    1         +
  .68   + 3       +         +         +    2    1         +
  .70   + 3       +         +         +    2    1         +
  .72   + 3       +         +         +    2   1+         +
  .74   + 3       +         +         +    2   1+         +
  .76   + 3       +         +         +    2   1+         +
  .78   +:3: : : :+: : : : :+: : : : :+: : 2 :1:+: : : : :+
  .80   + 3       +         +         +     2 1 +         +
  .82   + 3       +         +         +     2 1 +         +
  .84   + 3       +         +         +     2 1 +         +
  .86   + 3       +         +         +     2 1 +         +
  .88   +3        +         +         +     2 1 +         +
  .90   +3        +         +         +     21  +         +
  .92   +3        +         +         +     21  +         +
  .94   +3        +         +         +     21  +         +
  .96   +3        +         +         +     21  +         +
  .98   +3 : : : :+: : : : :+: : : : :+: : :21 :+: : : : :+
 1.00   +3        +         +         +     21  +         +
SCALE  -.1000    .2000     .5000     .8000     1.1000    1.4000
```

Figure V-51. CGT/PI-3 for QWS-8

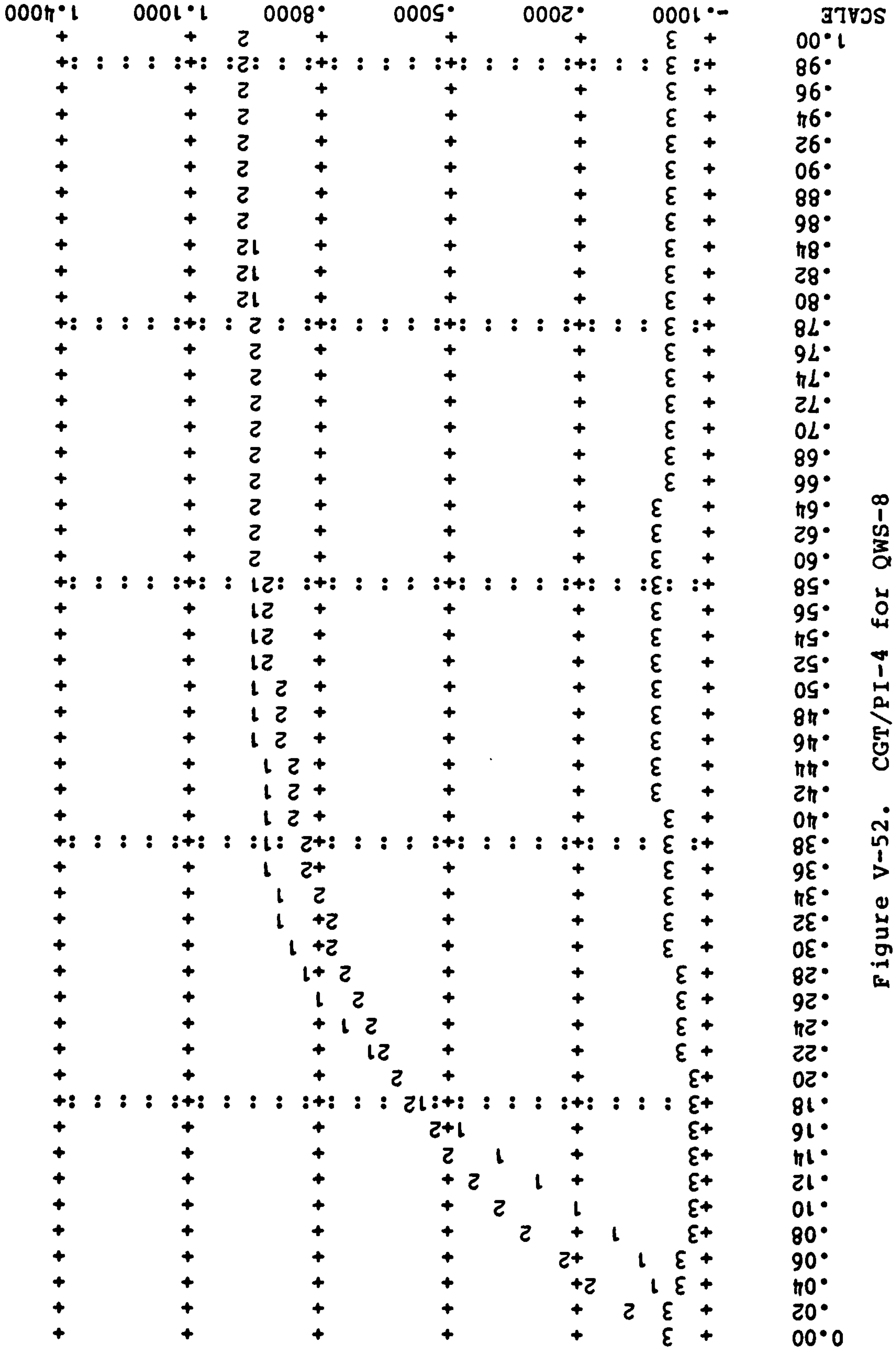


Figure V-52. CGT/PI-4 for QWS-8

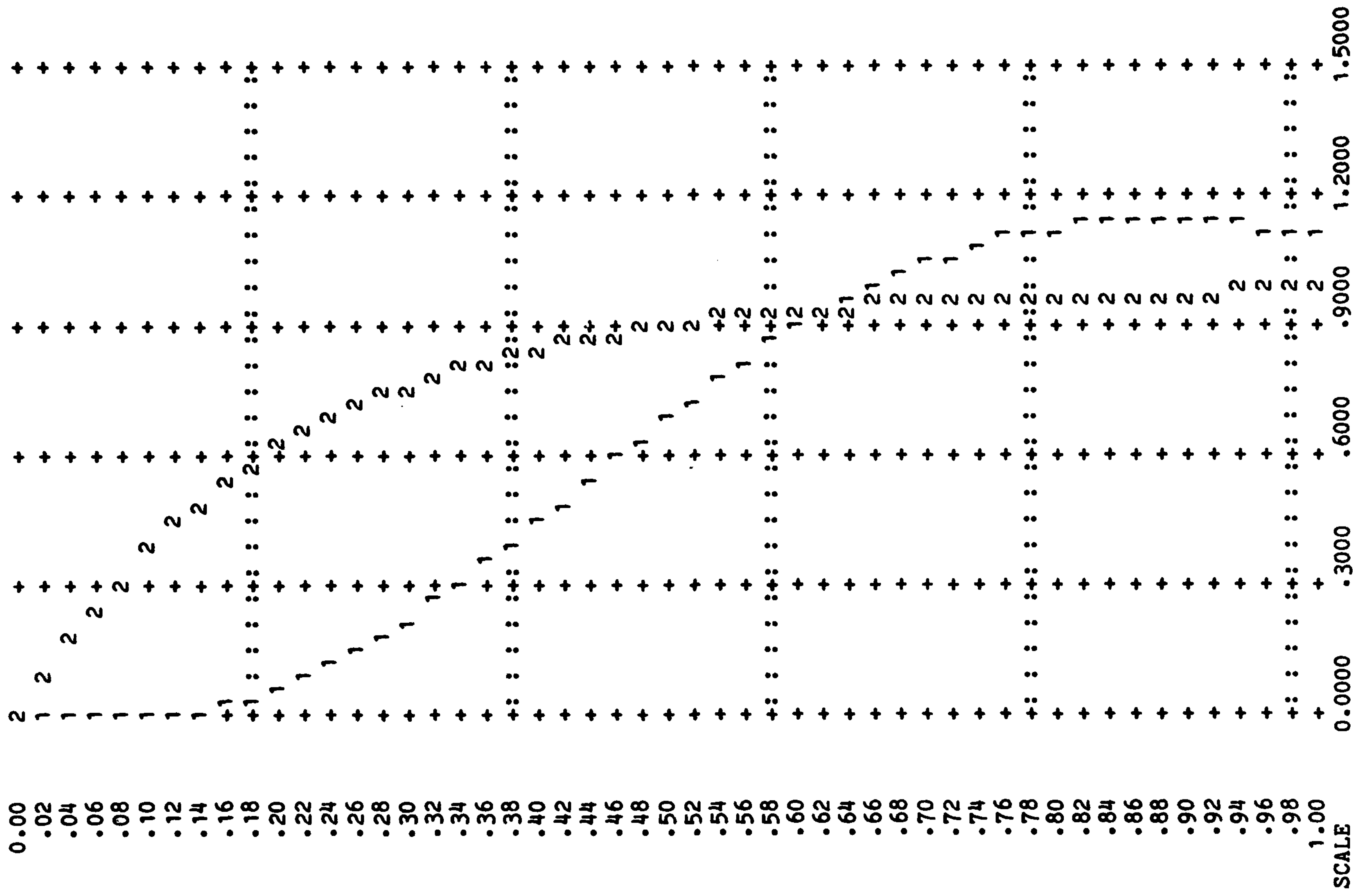


Figure V-53. Simple Design CGT (no disturbances, nominal truth model, command model 1)

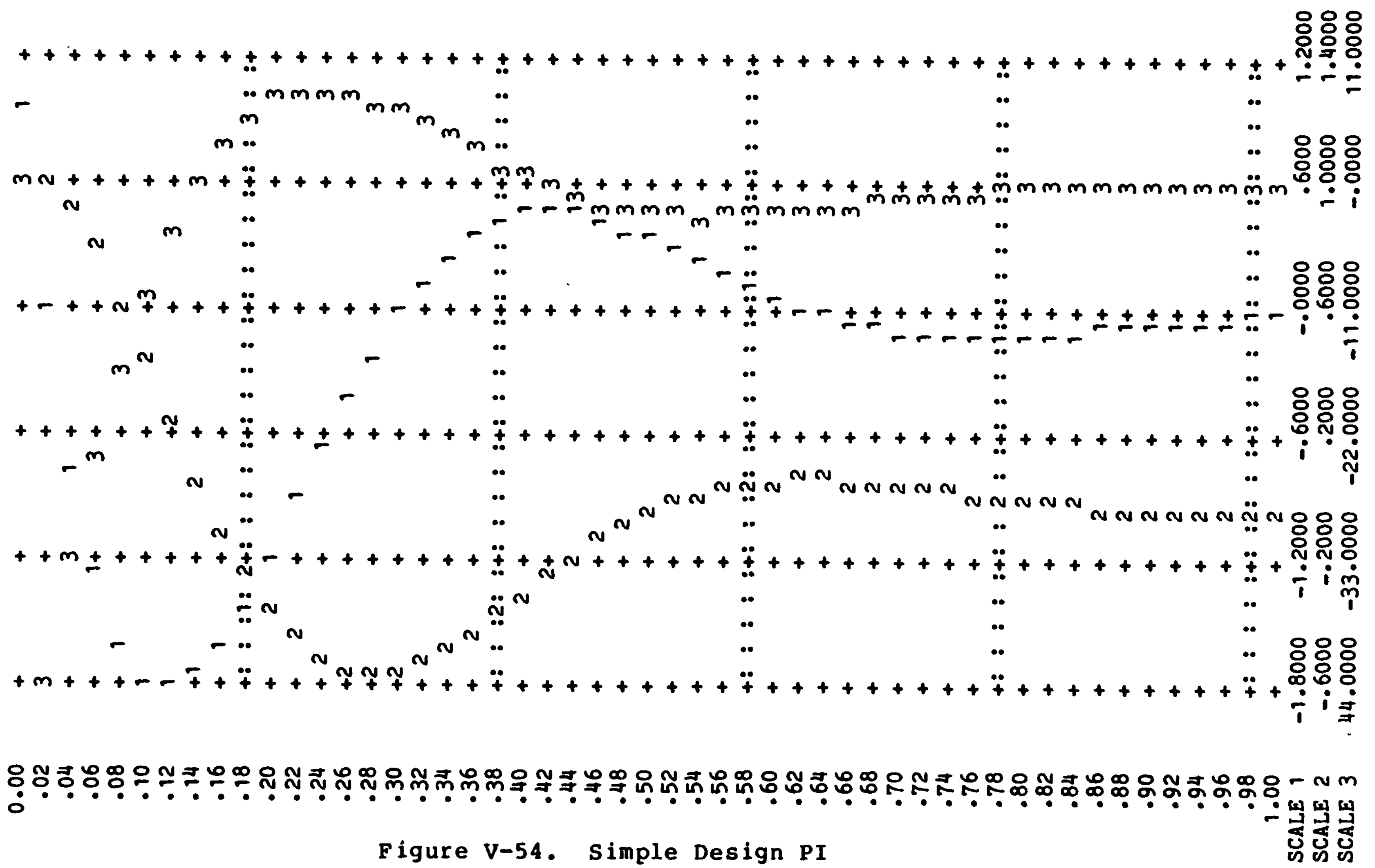


Figure V-54. Simple Design PI

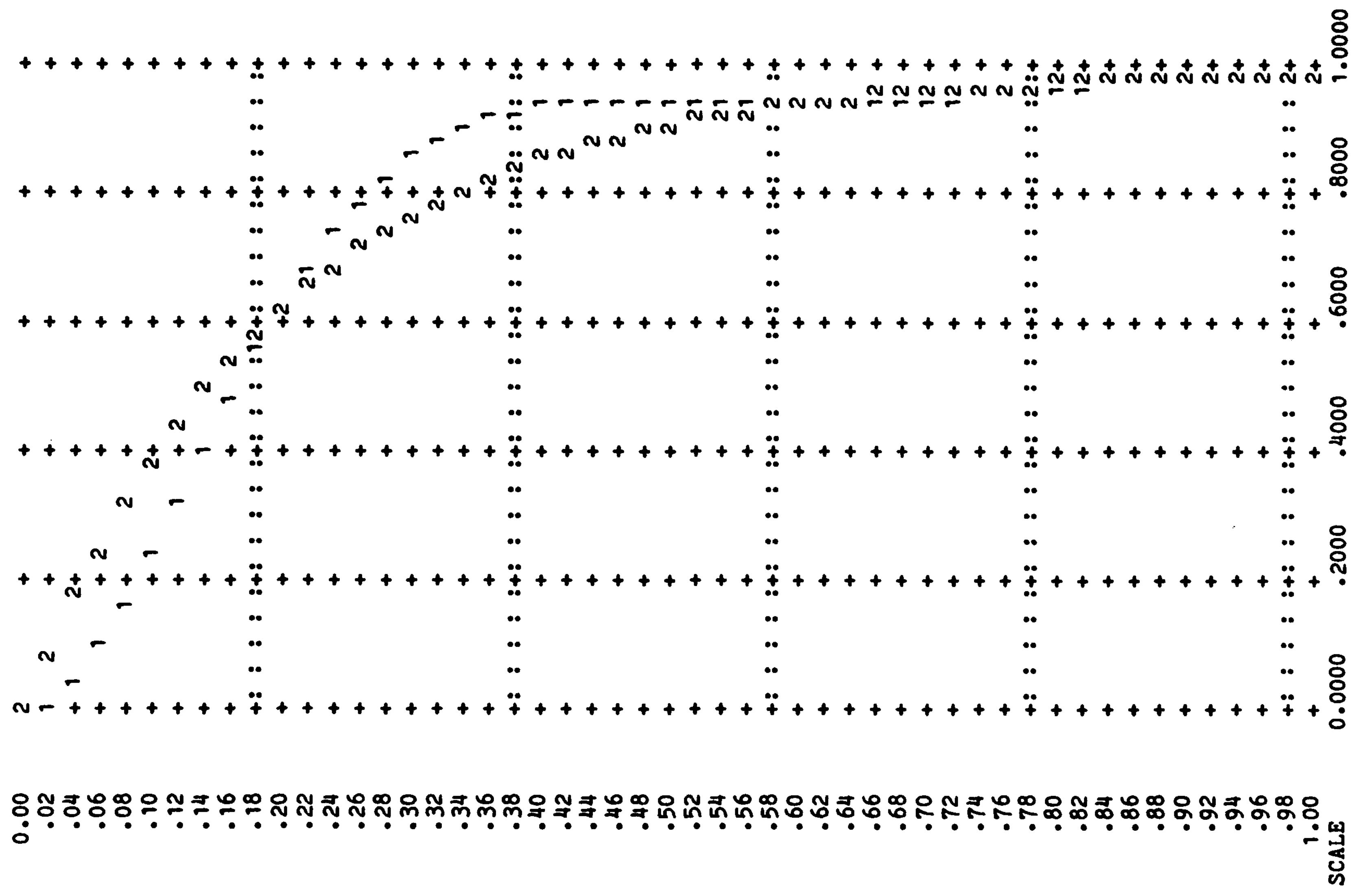


Figure V-55. Simple Design CGT/PI (no disturbances, nominal truth model, command model 1)

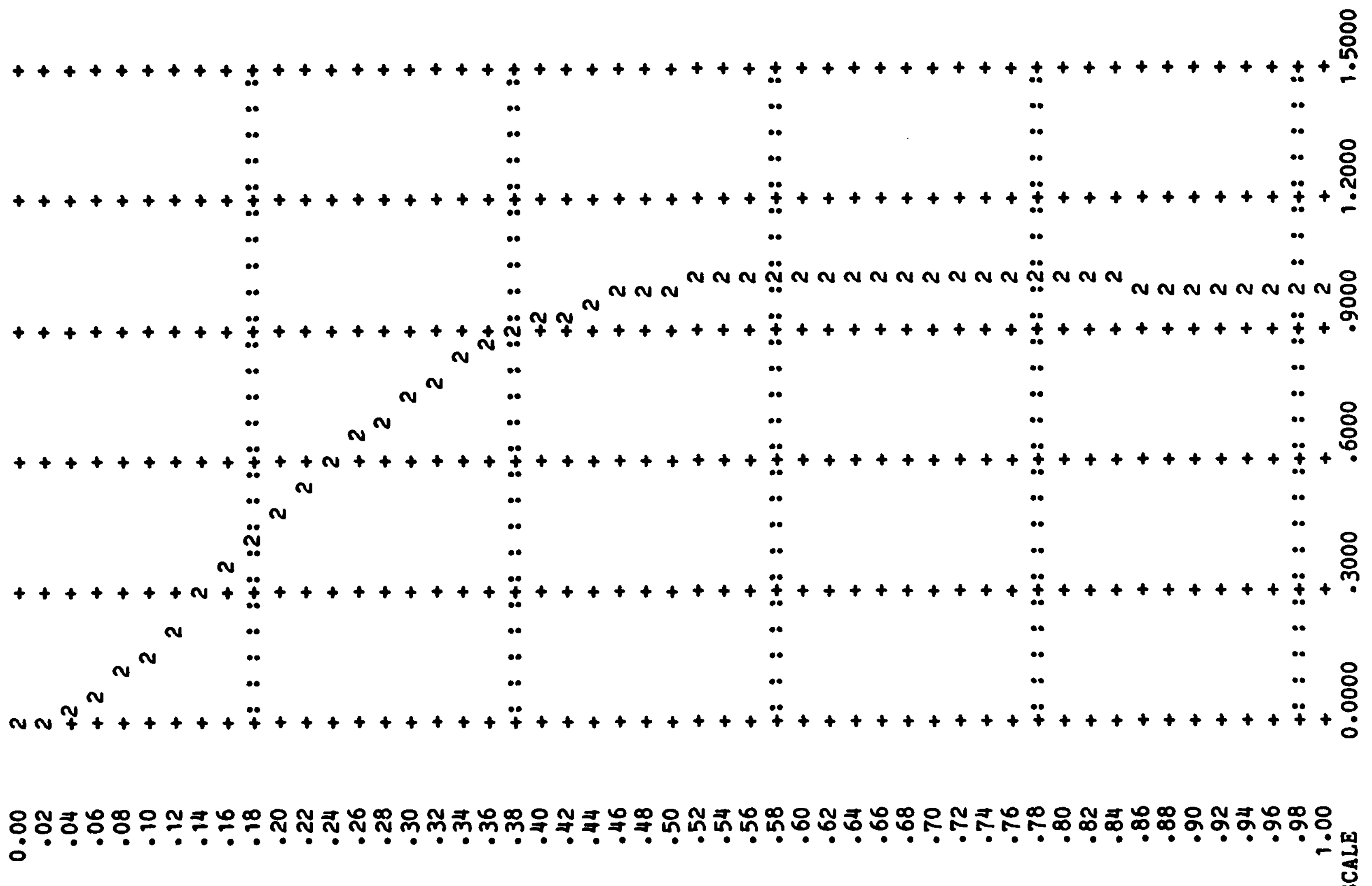


Figure V-56. Simple Design CGT/PI (no disturbances, nominal truth model, command model 2)

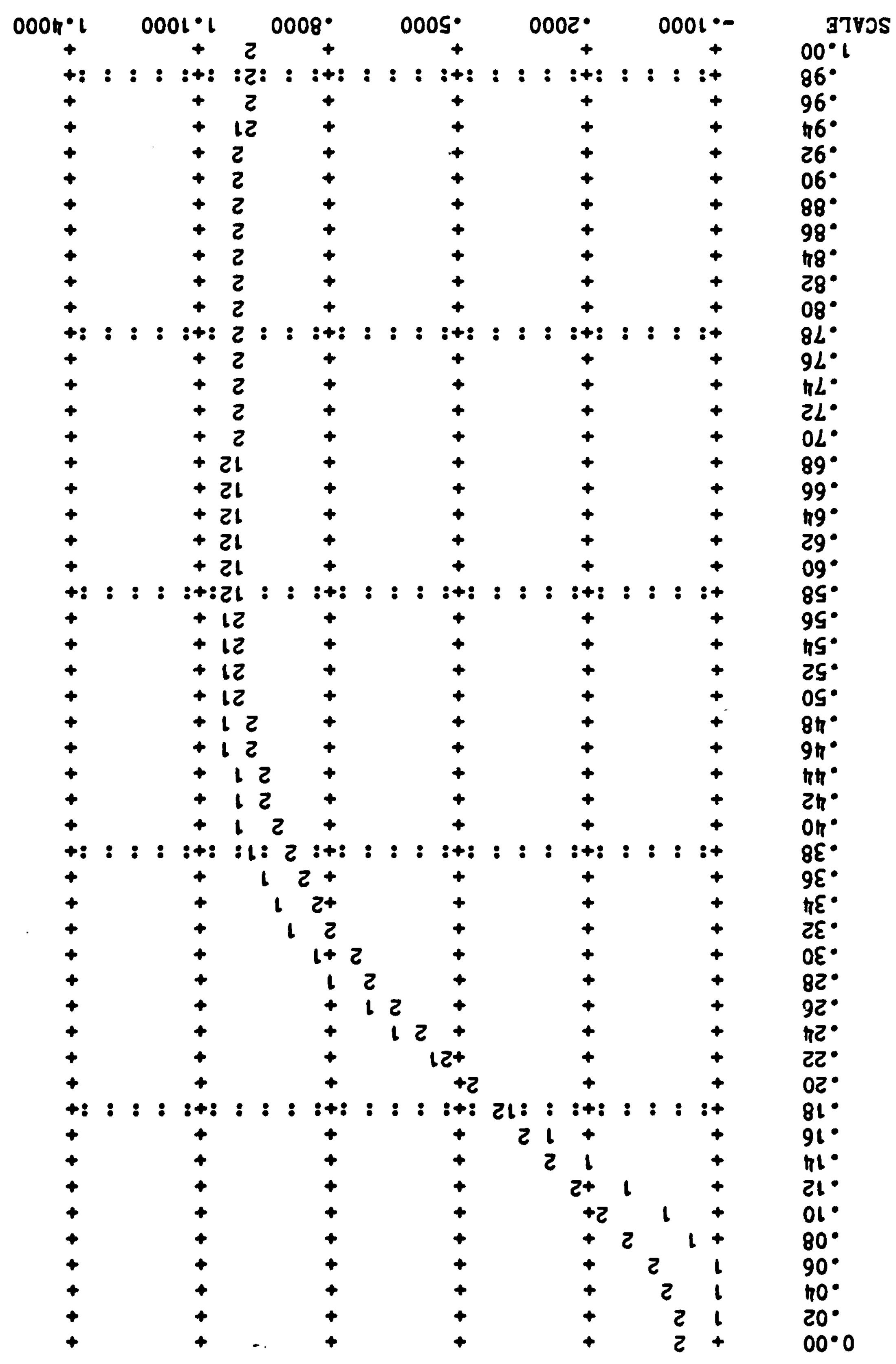


Figure V-57. Simple Design CGT/PI (constant disturbances, nominal truth model, command model 2)

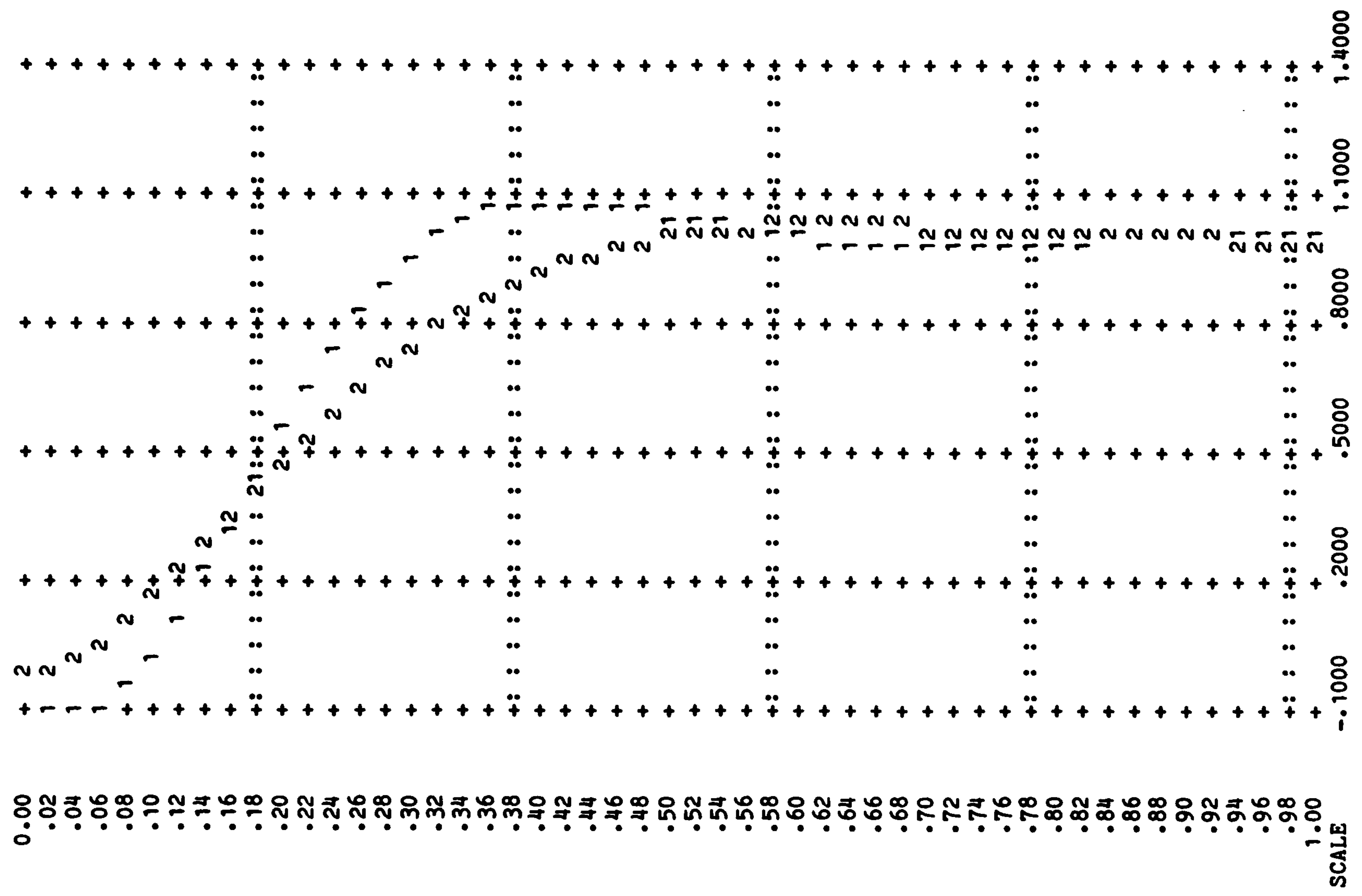


Figure V-58. Simple Design CGT/PI (constant disturbances, alternate truth model, command model 2)

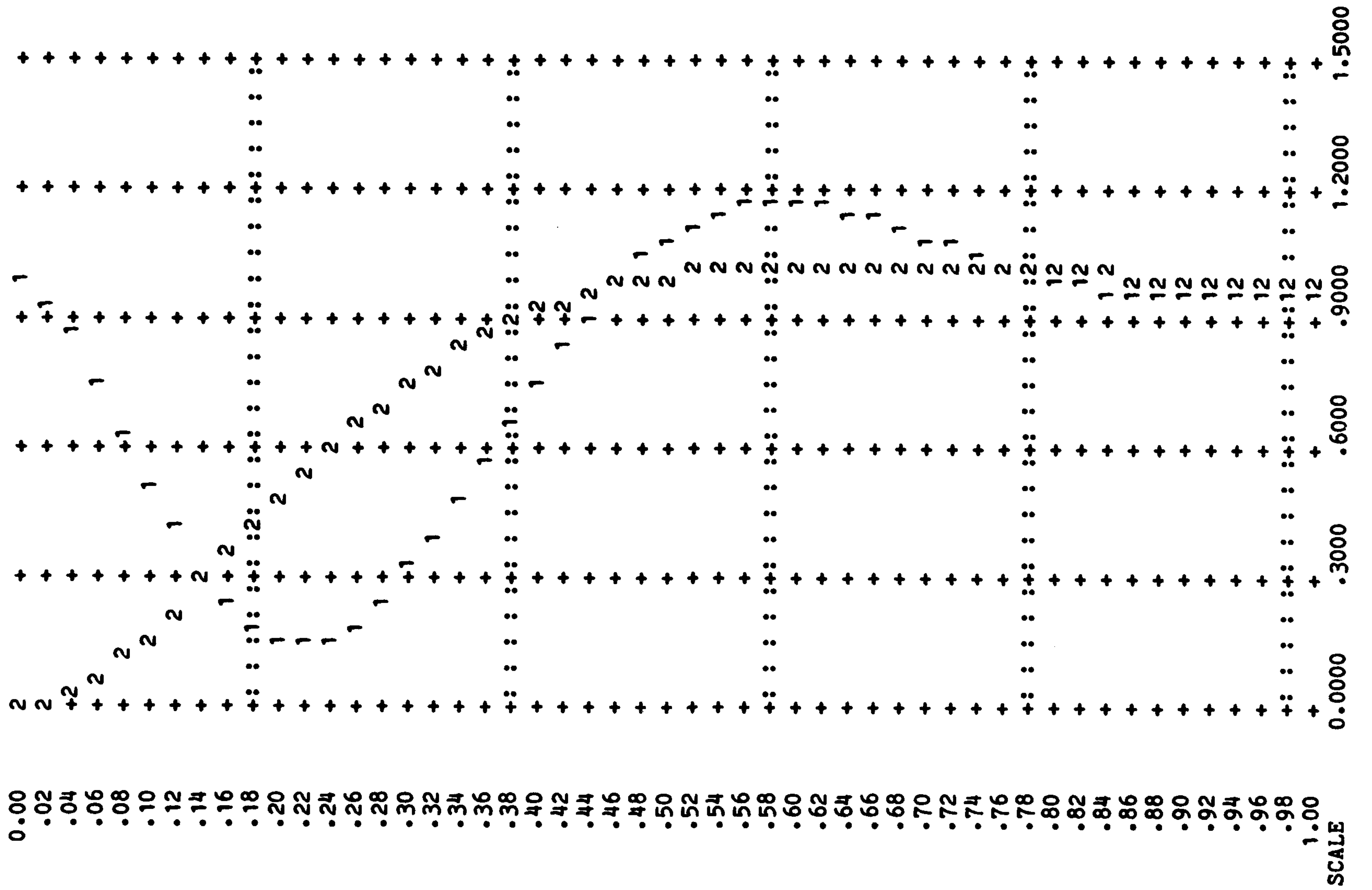


Figure V-59. Simple Design CGT/PI (constant disturbances, alternate truth model, initial conditions, command model 2)

VI. Summary and Recommendations

6.1 Summary

The first primary objective of this thesis has been to develop an interactive, user-oriented computer program similar to "CGTPIF" (Refs 9 and 21), which will also interface with "PFEVAL" (Ref 21), to aid in the design of CGT/PI/KF controllers, basing the design of the PI controller on the integral of the regulation error. Secondly, the design program was to be applied to an aircraft flight control problem to evaluate characteristics of this PI controller design, to compare the different CGT/PI designs within CGTPIQ, and to compare CGTPIQ with CGTPIF.

Chapter II introduced the CGT/PI/KF controller concept by briefly summarizing what is currently implemented in the interactive computer programs "CGTPIF" and "PFEVAL". The basic structure of the controller and the development of the necessary equations were presented.

Chapter III presented the theoretical development of the alternate PI and CGT/PI controller based on the integral of the regulation error. Four different formulations were presented and discussed for this CGT/PI controller. Each of these CGT/PI controllers were eventually implemented in code and compared.

Chapter IV discussed the CGT/PI/KF design computer program that was developed and gave the models that were employed in the design.

Chapter V presented and compared each of the PI and CGT/PI controllers discussed in Chapters II and III through use of the programs which were developed to implement them. General comments were made as to the usefulness of each program.

Appendix A is the CGTPIQ programmer's guide and contains a discussion of only those subroutines that were changed in "CGTPIF" to produce "CGTPIQ".

Appendix B is a complete, stand-alone, user's guide that contains all the information needed by the user of "CGTPIQ" to run the program successfully.

Appendix C is a complete listing of the program "CGTPIQ" as well as a listing of the subroutines that were changed for each of the CGT/PI formulations presented in Chapter III.

Appendix D contains a discussion of the CGT/PI/KF design evaluation equations that are used in "CGTPIQ".

6.2 Recommendations

During the development of the alternate PI controller equations, concepts such as better anti-windup capabilities due to separate proportional and integral gain channels and implicit model following were presented. As a follow-on study, these concepts could be thoroughly investigated using the program "CGTPIQ". Also, synthesis and analysis of complete CGT/PI/KF controllers (i.e. with

Kalman filters in the loop rather than assuming full state feedback) using the programs "CGTPIQ" and "PFEVAL" could be performed, to include: robustness characteristics, actuator saturation and associated windup phenomenon with a PI controller, reduced order design models (particularly actuator models and neglected bending/aeroelastic effects), and different equlibrium points for expansion of the linearized models. A similar study by Lt. Jean Howey using the programs "CGTPIF" and "PFEVAL" is currently underway ("Robust Flight Controllers". MS Thesis. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, December, 1983). It would also be useful to explore more fully the relative benefits of the two different types of PI designs embodied in "CGTPIQ" vs "CGTPIF", considering ease of achieving useful designs, capabilities to enhance robustness of the full-state feedback controller as well as CGT/PI/KF controllers, and other pertinent issues. As an additional area of endeavor, the two programs, "CGTPIF" and "CGTPIQ", could be combined into one program using the concepts presented in Ref (18) for a PIF controller. It would also be worthwhile for the entire man-machine interface (i.e., put the pilot in the loop with the CGT/PI/KF controller) to be implemented in a computer program to understand the inter-relationships of the overall system.

Bibliography

1. Barraud, A. Y. "A Numerical Algorithm to Solve $A^TXA-A=Q$," IEEE Trans. Automatic Control, AC-22 : 883-885, Oct 1977.

2. Broussard, J. R. "Command Generator Tracking," TASC TIM-612-3, The Analytic Sciences Corp., Reading, Massachusetts, March 1978.

3. Broussard, J. R. "Command Generator Tracking - The Discrete Time Case," TASC TIM-612-2, The Analytic Sciences Corp., Reading Massachusetts, March 1978.

4. Broussard, J.R., D.R. Downing and W.H. Bryant. "Design and Flight Testing of a Digital Optimal Control General Aviation Autopilot." 13th Congress of the International Council of the Aeronautical Sciences/AIAA Aircraft Systems and Technology Conference, Seattle, Washington, August, 1982.

5. Broussard, J. R., and M. G. Safanov. "Design of Generalized Discrete Proportional - Integral Controllers by Linear-Optimal Control Theory," TASC TIM-804-1, The Analytic Sciences Corp., Reading, Massachusetts, October, 1976.

6. Broussard, J. R., P. W. Berry, and R. F. Stengel. "Modern Digital Flight Control System Design For VTOL Aircraft," NASA CR-1159019. National Aeronautics and Space Administration, ~ Hampton, Virginia, March 1979.

7. CYBER Loader Version 1 Reference Manual. Publication number 60429800. Control Data Corporation, Sunnyvale, California, 1979.

8. Floyd, R. M. Aircraft Control Engineer. Unpublished AFTI/F-16 Aerodynamic Stability Derivative Data. Air Force Flight Dynamics Laboratory, Wright-Patterson AFB, Ohio, June 1980.

9. Floyd, Richard M. "Design of Advanaced Digital Flight Control Systems Via Command Generator Tracker (CGT) Synthesis Methods." Volumns 1 and 2. MS Thesis. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, December, 1981.

10. Govindaraj, K.S. and E.G. Rynaski. "Design Criteria for Optimal Flight Control Systems," AIAA Journal of Guidance and Control, Vol 3, No 4: 376-378, 1980.

11. Kalman, R. E., T. S. Englar, and R. S. Bucy. Fundamental Study of Adaptive Control Systems, ASD-TR-61-77, Wright-Patterson AFB, Ohio, 1961.

12. Kleinman, D. L., "A Description of Computer Programs Useful in Linear Systems Studies," Tec. Rep. TR-75-4. University of Connecticut, Storrs, Connecticut, October, 1975.

13. Kreindler, E., and D. Rothschild. "Model-Following in Linear-Quadratic Optimization," AIAA Journal, 14: 835-842, 1976.

14. Kuo, B. C. Digital Control Systems. Champaign, Illinois: SRL Publishing Company, 1979.

15. Kwakernaak, H., and R. Sivan. Linear Optimal Control Systems. New York: Wiley, 1972.

16. McRuer, D., I. Ashkenas, and D. Graham. Aircraft Dynamics and Automatic Control. Princeton, New Jersey: Princeton University Press, 1973.

17. Maybeck, Peter S., Stochastic Models, Estimation and Control, Vol 1. New York: Academic Press, 1979.

18. Maybeck, Peter S., Stochastic Models, Estimation and Control, Vol 3. New York: Academic Press, 1982.

19. Maybeck, Peter S., Richard M. Floyd and Alphronzo Moseley. "Synthesis and Performance Evaluation Tools for CGT/PI Advanced Digital Flight Control Systems." IEEE NAECON, May, 1983.

20. Miller, William G. "Robust Multivariable Controller Design Via Implicit Model Following Methods". MS Thesis. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, December, 1983.

21. Moseley, Alphronzo. "Design of Advanced Digital Flight Control Systems Via Command Generator Tracker (CGT) Synthesis Methods." Volumns 1 and 2. MS Thesis. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, December, 1982.

Appendix A

CGTPIQ Programmer's Guide

A.1 Introduction

CGTPIQ is a controller design program in which the PI controller is based on the integral of the regulation error. The program is designed to execute interactively. Three design paths are offered: (1) design of a Proportional-plus-Integral (PI) controller via linear-quadratic (LQ) methodology using implicit and/or explicit quadratic weights; (2) design of a Command Generator Tracker, either open-loop (CGT) or closed-loop (CGT/PI); and (3) design of a Kalman filter (KF). These three designs are components of a final controller implemented as a Command Generator Tracker, with an inner-loop proportional-plus-integral controller, and a Kalman filter for state estimation (CGT/PI/KF). For each design path there is a corresponding set of routines to evaluate the quality of the design achieved (see Appendix D).

The program is written in FORTRAN IV and consists of more than 2900 lines of source code. In addition, numerous routines are employed from a library of matrix routines described in Ref (12). Since the resulting program is large both in code and in memory utilization for array storage, direct and complete loading of the program exceeds memory limits for interactive execution on the ASD CYBER computer system. A technique referred to as "segmentation" is employed to provide selective loading of routines during execution so that memory usage remains within the limits for interactive execution.

This guide is a brief discussion of the subroutines that were changed in converting the program written by Floyd (Ref 9), as modified by Moseley (Ref 21), called "CGTPIF" to the program called "CGTPIQ" which bases the PI controller on the integral of the regulation error instead of the control difference or "pseudorate" as done in "CGTPIF". The equations presented in this guide reflect the development presented in Chapter III (only CGT/PI formulation 1, Section 3.3.2.1 is presented in detail). Since the two programs are exactly alike except for a few subroutines, it is felt that repeating the entire development presented in Floyd's and Moseley's theses would not be required by most users. If the user wishes to understand the entire program development he should obtain copies of Floyd's and Moseley's theses (Refs 9 and 21), and taking their development along with the changes presented in this guide, a complete description of the program will be available.

In converting CGTPIF to CGTPIQ major changes were made to subroutines CDIF, SREGPI, WXUS, FORMX (name also changed to FORMXU), CGTKX, and MODIFX (name also changed to MODXU) with minor changes made to MAIN, CGTXQ, UCGT, and SCMD. There were also numerous cosmetic changes made so that the program elements sounded like what they were actually representing. All of these changes will be presented in the following sections.

A.2 Program Changes

A.2.1 CDIF

'CDIF' is called by the subroutine 'PIMTX'. 'CDIF' sets up two augmented matrices for the integral of the regulation error PI controller. These matrices are:

$$\underline{\Phi}_{\int} = \begin{bmatrix} \underline{\Phi} & \underline{0} \\ \underline{C} & \underline{I} \end{bmatrix} \qquad \text{(A-1a)}$$

$$\underline{B}_{\int} = \begin{bmatrix} \underline{B}_d \\ \underline{D}_y \end{bmatrix} \qquad \text{(A-1b)}$$

$\underline{\Phi}_{\int}$ and $\underline{B}_{\int}$ are stored in vector CTL of /CONTROL/.

A.2.2 SREGPI, WXUS, and FORMXU

Computations involved in the design of the PI controller are directed by routine SREGPI. Routine 'WXUS' is called first to determine the quadratic weighting matrices of the discrete-time optimal cost function from the continuous-time input quadratic weights. Quadratic weighting matrices (assumed diagonal) are entered directly by the user from the terminal for costs assigned to output, integral of the regulation error, and control magnitudes--$\underline{Y}$, $\underline{U}_q$, and $\underline{U}_y$ respectively. These matrices are stored in vector "RPI" of /CREGPI/. An augmented perturbation state vector is defined to be

$$\underline{x} = \begin{bmatrix} \int \underline{x} \\ \int \underline{q} \end{bmatrix} \quad \text{(A-2)}$$

A weighting matrix on the state vector $\underline{x}$ is formed as

$$\underline{X}_{c11} = \underline{C}^T \underline{Y} \underline{C} \quad \text{(A-3a)}$$

$$\underline{X}_{c22} = \underline{U}_q \quad \text{(A-3b)}$$

$$\underline{X}_{c12} = \underline{0} \quad \text{(A-3c)}$$

Routine 'FORMXU' performs these computations and forms

$$\underline{XU} = \begin{bmatrix} \underline{X} & \underline{S}_c \\ \underline{S}_c{}^T & \underline{U}_c \end{bmatrix} \quad \text{(A-4)}$$

where

$$\underline{X} = \begin{bmatrix} \underline{X}_{c11} & \underline{X}_{c12} \\ \underline{X}_{c12}{}^T & \underline{X}_{c22} \end{bmatrix} \quad \text{(A-5a)}$$

$$\underline{U}_c = \underline{U}_y + \underline{D}_y{}^T \underline{Y} \underline{D}_y \quad \text{(A-5b)}$$

$$\underline{S}_c = \begin{bmatrix} \underline{C}^T \underline{Y} \underline{D}_y \\ \underline{0} \end{bmatrix} \quad \text{(A-5c)}$$

The user is then given the opportunity to modify individual elements of $\underline{X}$ (symmetry is preserved automatically by WXUS), as for instance, to alter individual diagonal elements of

$\underline{X}_{c11}$. The associated continuous-time cost function is,

$$J_c = \frac{1}{2}\int_{t_0}^{t_{N+1}} \begin{bmatrix}\underline{x}(t)\\ \underline{u}(t)\end{bmatrix}^T \begin{bmatrix}\underline{X}_c & \underline{S}_c\\ \underline{S}_c^T & \underline{U}_c\end{bmatrix} \begin{bmatrix}\underline{x}(t)\\ \underline{u}(t)\end{bmatrix} dt \tag{A-6}$$

where

$$\underline{x} = \begin{bmatrix}\delta\underline{x}\\ \delta\underline{q}\end{bmatrix} \tag{A-6a}$$

$$\underline{u} = \delta\underline{u} \tag{A-6b}$$

$$\underline{X}_c = \begin{bmatrix}\underline{X}_{c11} & \underline{X}_{c12}\\ \underline{X}_{c12}^T & \underline{X}_{c22}\end{bmatrix} \tag{A-6c}$$

where $\underline{q}$ is the integral of the regulation error. The corresponding discrete-time cost function is defined by:

$$J = \sum_{i=0}^{N} \tfrac{1}{2}[\underline{x}^T(t_i)\underline{X}_\delta\underline{x}(t_i) + \underline{u}^T(t_i)\underline{U}_\delta\underline{u}(t_i) + 2\underline{x}^T(t_i)\underline{S}_\delta\underline{u}(t_i)] \tag{A-7}$$

and the discrete costs are,

$$\underline{X}_\delta = \int_{t_i}^{t_i+T} [\underline{\Phi}_\delta^T(\tau)\underline{X}_c\underline{\Phi}_\delta(\tau)]d\tau \tag{A-8a}$$

$$\underline{U}_\delta = \int_{t_i}^{t_i+T} [\underline{B}_\delta^T(\tau)\underline{X}_c\underline{B}_\delta(\tau) + \underline{U}_c + \underline{B}_\delta^T(\tau)\underline{S}_c + \underline{S}_c^T\underline{B}\ (\tau)]d\tau \tag{A-8b}$$

$$\underline{S}_\delta = \int_{t_i}^{t_i+T} [\underline{\Phi}_\delta^T(\tau)\underline{X}_c\underline{B}_\delta(\tau) + \underline{\Phi}_\delta^T(\tau)\underline{S}_c]d\tau \tag{A-8c}$$

in which

$$\underline{\Phi}_{\delta}(\tau) = \begin{bmatrix} \underline{\Phi}(\tau) & \underline{0} \\ \underline{C} & \underline{I} \end{bmatrix} \qquad \text{(A-9a)}$$

$$\underline{B}_{\delta}(\tau) = \begin{bmatrix} \underline{B}_d(\tau) \\ \underline{D}_y \end{bmatrix} \qquad \text{(A-9b)}$$

$$\underline{B}_d(\tau) = \int_0^{\tau} \underline{\Phi}(\sigma)\underline{B}d\sigma \qquad \text{(A-9c)}$$

The integrals in Equations (A-8a,b,c) are approximated in a two-step computation. First, $\underline{\Phi}_{\delta}$ and $\underline{B}_{\delta}$ are treated as constants over the sample interval with value set to their respective averaged values at the beginning and end of the interval:

$$\bar{\underline{\Phi}}_{\delta} = \tfrac{1}{2}[\underline{I} + \underline{\Phi}_{\delta}] \qquad \text{(A-10a)}$$

$$\bar{\underline{B}}_{\delta} = \tfrac{1}{2}[\underline{0} + \underline{B}_{\delta}] \qquad \text{(A-10b)}$$

(where the overbar means approximation) in which $\underline{\Phi}_{\delta}$ and $\underline{B}_{\delta}$ are as defined in Equations (A-1a,b). With these approximations, each of the integrands is constant over the integration time T, so the integrals are obtained as

$$\bar{\underline{X}}_{\delta} = T[\bar{\underline{\Phi}}_{\delta}^{T}\underline{X}\bar{\underline{\Phi}}_{\delta}] \qquad \text{(A-11a)}$$

$$\bar{\underline{U}}_{\delta} = T[\bar{\underline{B}}_{\delta}{}^{T}\underline{X}\bar{\underline{B}}_{\delta} + \underline{U}_c + \bar{\underline{B}}_{\delta}{}^{T}\underline{S}_c + \underline{S}_c{}^{T}\bar{\underline{B}}_{\delta}] \qquad \text{(A-11b)}$$

$$\bar{\underline{S}}_{\delta} = T[\bar{\underline{\Phi}}_{\delta}^{T}\underline{X}\bar{\underline{B}}_{\delta} + \bar{\underline{\Phi}}_{\delta}^{T}\underline{S}_c] \qquad \text{(A-11c)}$$

This provides a better approximate evaluation than simple Euler integration provides. These three discrete-time costs are returned by 'WXUS' as arguments "X", "U", and "S", respectively.

Note that the cross weighting matrix $\bar{\underline{S}}_\delta$ has been introduced into the cost function by the discretization process and will generally be nonzero even if $\underline{S}_c$ in Equation (A-6) were zero. This cost function is seen to be in the standard form used previously and all development can proceed as if the problem were a discrete problem and not a continuous one.

Now, in order to obtain an equivalent discrete-time cost function with no cross weighting (to allow use of standard Riccati equation solvers as in Ref (12) that assume such a form), routine 'PXUP' is called to compute a modified system and weighting matrices. Define a new system (Ref 14)

$$\underline{x}(t_{i+1}) = \bar{\underline{\Phi}}_\delta{}' \underline{x}(t_i) + \bar{\underline{B}}_\delta \underline{u}'(t_i) \tag{A-12}$$

for which

$$\bar{\underline{\Phi}}_\delta{}' = \bar{\underline{\Phi}}_\delta - \bar{\underline{B}}_\delta \bar{\underline{U}}_\delta{}^{-1} \bar{\underline{S}}_\delta{}^T \tag{A-13a}$$

and

$$\underline{u}'(t_i) = \underline{u}(t_i) + \bar{\underline{U}}_\delta{}^{-1} \bar{\underline{S}}_\delta{}^T \underline{x}(t_i) \tag{A-13b}$$

for which the corresponding cost function becomes

$$J' = \sum_{i=0}^{N} \tfrac{1}{2} [\underline{x}^T(t_i) \bar{\underline{X}}_\delta{}' \underline{x}(t_i) + \underline{u}'^T(t_i) \bar{\underline{U}}_\delta{}' \underline{u}'(t_i)] \tag{A-14}$$

with

$$\underline{\bar{X}}_\delta' = \underline{\bar{X}}_\delta - \underline{\bar{S}}_\delta \underline{\bar{U}}_\delta^{-1} \underline{\bar{S}}_\delta^T \tag{A-15}$$

The cost function of Equation (A-14) is now in standard form for solution of the steady-state Riccati equation. PXUP returns matrices $\underline{\Phi}_\delta'$, $\underline{\bar{X}}_\delta'$, and $\underline{\bar{U}}_\delta^{-1}\underline{\bar{S}}_\delta^T$ of Equations (A-13a), (A-15), and (A-13b), respectively. An additional matrix needed for the routine which computes the solution to the Riccati equation is also computed by PXUP:

$$\underline{\bar{U}}_\delta' = \underline{\bar{B}}_\delta \underline{\bar{U}}_\delta^{-1} \underline{\bar{B}}_\delta^T \tag{A-16}$$

SREGPI next computes the steady-state solution to the discrete-time Riccati equation using routine 'DRIC' of LIBRARY. DRIC solves for $\underline{K}_R$ in

$$\underline{K}_R = \underline{\Phi}_\delta'^T \underline{K}_R (\underline{I} + \underline{\bar{U}}_\delta' \underline{K}_R)^{-1} \underline{\Phi}_\delta' + \underline{\bar{X}}_\delta' \tag{A-17}$$

using an iterative procedure discussed in Ref (12). In addition to $\underline{K}_R$, DRIC returns the closed-loop system matrix

$$\underline{\Phi}_{\delta CL} = (\underline{I} + \underline{\bar{U}}_\delta' \underline{K}_R)^{-1} \underline{\Phi}_\delta' \tag{A-18}$$

which is stored in vector RPI.

Routine 'GCSTAR' then is called to compute the optimal feedback gain matrix for the original system. The optimal feedback gains for the modified system of Equation

(A-12) are,

$$\underline{G}_c^{*\prime} = [\underline{\bar{U}}_{\mathcal{S}} + \underline{\bar{B}}_{\mathcal{S}}^T \underline{K}_R \underline{\bar{B}}_{\mathcal{S}}]^{-1} \underline{\bar{B}}_{\mathcal{S}}^T \underline{K}_R \underline{\bar{\Phi}}_{\mathcal{S}}' \tag{A-19}$$

and from these the optimal feedback gains for the original system are

$$\underline{G}_c^{*} = \underline{G}_c^{*\prime} + \underline{\bar{U}}_{\mathcal{S}}^{-1} \underline{\bar{S}}_{\mathcal{S}}^T \tag{A-20}$$

Remembering from Equation (III-25) that

$$\underline{G}_c^{*} = [\underline{G}_{c1}^{*} \quad \underline{G}_{c2}^{*}] \tag{A-21}$$

the calculation beginning with a continuous-time cost description will produce the correct $\underline{G}_c^{*}$. The only question that might arise is whether the substitution of $\underline{K}_R'$ for $\underline{K}_c$ is a valid substitution. This will be shown in the last section of this guide. The PI controller in incremental form utilizing these gains is implemented as

$$\begin{aligned}\underline{u}^{*}(t_i) = \underline{u}^{*}(t_{i-1}) - \underline{G}_{c1}^{*}[\underline{x}(t_i) - \underline{x}(t_{i-1})] \\ - \underline{G}_{c2}^{*} \; [\underline{C} \quad \underline{D}_y] \begin{bmatrix} \underline{x}(t_{i-1}) \\ \underline{u}(t_{i-1}) \end{bmatrix}\end{aligned} \tag{A-22}$$

The controller evaluation routines which propagate the response of the PI controlled system to non-zero initial conditions use Equation (A-22) to compute the control input. Note that this assumes that the outputs are to be driven to zero by the PI controller. No provision is made for evaluation of the PI controller in response to control inputs, therefore, $\underline{y}_d$ will be zero for all time. Thus the matrix $\underline{E}$ is not required in the computation, but is computed and saved in vector LEE of /LREGPI/ so that a comparison can be made between the proportional and integral channels of the PI controller according to

$$\underline{K}_p = [\underline{I} - \underline{E}\underline{D}_y]^{-1}\underline{E} \tag{A-23a}$$

$$\underline{K}_i = [\underline{I} - \underline{E}\underline{D}_y]^{-1}\underline{G}_{c2}^* \tag{A-23b}$$

(These are computed at the end of SREGPI.) and since a major portion of $\underline{E}$ known as the $\underline{L}$ matrix where

$$\underline{L} = \underline{G}_{c1}^* - \underline{G}_{c2}^* \underline{K}_{c22}^{-1} \underline{K}_{c12}^T \tag{A-23c}$$

is needed in the closed-loop CGT/PI controller development. The $\underline{L}$ matrix is saved in vector LEL of /LREGPI/. (See Section 3.3.2 for changes for CGT/PI formulations 2,3,4.)

A.2.3 CGTKX and SCMD

SCGT directs the computations involved in design of an open-loop CGT or closed-loop CGT/PI controller. SCGT calls routine CGTKX to compute the gains employed by the CGT and CGT/PI controllers. For the open-loop CGT controller routine SCMD sets matrices $\underline{G}_{c1}^*$ and $\underline{G}_{c2}^*$ to zero. CGTKX computes gains on command model states and inputs and disturbance states, respectively, as

$$\underline{K}_{xm} = \underline{L}\underline{A}_{11} + \underline{A}_{21} \qquad \text{(A-24a)}$$

$$\underline{K}_{xu} = \underline{L}\underline{A}_{12} + \underline{A}_{22} \qquad \text{(A-24b)}$$

$$\underline{K}_{xn} = \underline{L}\underline{A}_{13} + \underline{A}_{23} \qquad \text{(A-24c)}$$

These three gains are stored in vector CGT. (See Section 3.3.2 for changes for CGT/PI formulations 2,3,4.)

The closed-loop CGT/PI control law is implemented in incremental form as

$$\begin{aligned}\underline{u}^*(t_i) = \underline{u}^*(t_{i-1}) &- \underline{G}_{c1}{}^*[\underline{x}(t_i) - \underline{x}(t_{i-1})] \\ &+ \underline{K}_{xm}[\underline{x}_m(t_i) - \underline{x}_m(t_{i-1})] \\ &+ \underline{K}_{xu}[\underline{u}_m(t_i) - \underline{u}_m(t_{i-1})] \\ &+ \underline{K}_{xn}[\underline{n}_d(t_i) - \underline{n}_d(t_{i-1})] \\ &+ \underline{G}_{c2}{}^*\left\{[\underline{C}_m \ \underline{D}_m]\begin{bmatrix}\underline{x}_m(t_{i-1}) \\ \underline{u}_m(t_i)\end{bmatrix} - [\underline{C} \ \underline{D}_y]\begin{bmatrix}\underline{x}(t_{i-1}) \\ \underline{u}(t_{i-1})\end{bmatrix}\right\}\end{aligned} \qquad \text{(A-25)}$$

(See Section 3.3.2 for changes for CGT/PI formulations 2,3,4.) The open-loop CGT is obtained by employing Equation (A-25) with PI gains $\underline{G}_{c1}{}^*$ and $\underline{G}_{c2}{}^*$ both zero matrices, giving the effective result for the open-loop CGT control law as

$$\begin{aligned}\underline{u}^*(t_i) = \underline{u}^*(t_{i-1}) &+ \underline{A}_{21}[\underline{x}_m(t_i) - \underline{x}_m(t_{i-1})] \\ &+ \underline{A}_{22}[\underline{u}_m(t_i) - \underline{u}_m(t_{i-1})] \\ &+ \underline{A}_{23}[\underline{n}_d(t_i) - \underline{n}_d(t_{i-1})]\end{aligned} \qquad \text{(A-26)}$$

Equation (A-25) is used by the controller evaluation routines to compute control inputs for either CGT controller.

A.2.4 UCGT

UCGT computes the control input due to the CGT controller alone and adds it to the control given by the PI controller from URPI. The increment due to the CGT or CGT/PI alone is added as

$$\begin{aligned} \underline{u}^*(t_i) \leftarrow \underline{u}^*(t_i) &+ \underline{K}_{xm}[\underline{x}_m(t_i) - \underline{x}_m(t_{i-1})] \\ &+ \underline{K}_{xu}[\underline{u}_m(t_i) - \underline{u}_m(t_{i-1})] \\ &+ \underline{K}_{xn}[\underline{n}_d(t_i) - \underline{n}_d(t_{i-1})] \\ &+ \underline{G}_{c2}^*[\underline{C}_m \quad \underline{D}_m] \begin{bmatrix} \underline{x}_m(t_{i-1}) \\ \underline{u}_m(t_i) \end{bmatrix} \end{aligned} \tag{A-27}$$

(See Section 3.3.2 for changes for CGT/PI formulations 2,3,4.)

A.2.5 MODXU

MODXU computes the combined implicit/explicit quadratic weights according to

$$\underline{\underline{XU}} = \begin{bmatrix} \underline{X}_c + \hat{\underline{X}}_I & \underline{0} & \underline{S}_c + \hat{\underline{S}}_I \\ \underline{0} & \underline{U}_q & \underline{0} \\ (\underline{S}_c + \hat{\underline{S}}_I)^T & \underline{0} & \underline{U}_c + \hat{\underline{U}}_I \end{bmatrix} \tag{A-28}$$

A.2.6 MAIN and CGTXQ

MAIN is changed by increasing the storage space allotted to common /CREGPI/ and vector NVRPI to 675 to take care of the two added matrices $\underline{L}$ and $\underline{E}$.

CGTXQ is changed by adding vector space for matrices $\underline{L}$ and $\underline{E}$ in Common /LREGPI/. These are called LEL and LEE, respectively.

A.2.7 Cosmetic Changes

All reference to the vector notation on the left has been changed to the notation on the right

KX	- GC1
KZ	- GC2
LKX	- LGC1
LKZ	- LGC2
LKXAxx	- LELAxx
RKX	- RGC1 (in subroutine URPI)
RKZ	- RGC2 (in subroutine URPI)
RKX	- REL (in subroutine CGTKX)

(See Appendix C for changes for CGT/PI formulations 2,3,4.)

A.3 Derivation Showing that $\underline{K}_R{}' = \underline{K}_C$

In the simple proof we know that the cost to be minimized in the case of both K's is given by (Ref 15)

$$\underline{x}^T \underline{K}_C \underline{x} = J_{min} = J_{min}' = \underline{x}^T \underline{K}_R{}' \underline{x} \qquad \text{(A-29)}$$

and since the states, the $\underline{x}$'s, are the same on both sides of

Equation (A-29), it follows directly that

$$\underline{K}_c = \underline{K}_R' \qquad \text{(A-30)}$$

It can also be shown rigorously that

$$\underline{K}_R' = \underline{K}_c = \underline{X} + \underline{\Phi}^T \underline{K}_c \underline{\Phi} - [\underline{B}_d^T \underline{K}_c \underline{\Phi} + \underline{S}^T]^T \underline{G}_c^* \qquad \text{(A-31)}$$

Given Equation (III-18a) which is the $\underline{K}_c$ portion of Equation (A-31), lets show that $\underline{K}_R'$ is equal to the same relationship. Starting with our equation in question, Equation (III-49), utilize Equation (III-51) to get (dropping the delta subscripts)

$$\underline{K}_R' = \underline{X}' + \underline{\Phi}'^T \underline{K}_R' \underline{\Phi}' - \underline{\Phi}'^T \underline{K}_R' \underline{B} \underline{G}_c^{*\prime} \qquad \text{(A-32)}$$

Substituting Equation (III-48b) for $\underline{X}'$, Equation (III-47b) for $\underline{\Phi}'$, Equation (III-52) for $\underline{G}_c^{*\prime}$, multiplying out the resultant equation and canceling like terms, we have

$$\begin{aligned}\underline{K}_R' = [\underline{X} + \underline{\Phi}^T \underline{K}_R' \underline{\Phi} - \underline{\Phi}^T \underline{K}_R' \underline{B} \underline{G}_c^*] - [\underline{S}\underline{U}^{-1} \{\underline{B}^T \underline{K}_R' \underline{\Phi} + \underline{S}^T\} \\ - \underline{S}\underline{U}^{-1} \underline{B}^T \underline{K}_R' \underline{B} \underline{G}_c^*]\end{aligned} \qquad \text{(A-33)}$$

Noting that the portion of Equation (A-33) in braces {} is equal to

$$\{\ \} = [\underline{U} + \underline{B}^T \underline{K}_R' \underline{B}] \underline{G}_c^* = \underline{U} \underline{G}_c^* + \underline{B}^T \underline{K}_R' \underline{B} \underline{G}_c^* \qquad \text{(A-34)}$$

and making this substitution, canceling the resultant like terms, we have

$$\underline{K}_R' = \underline{X} + \underline{\Phi}^T\underline{K}_R'\underline{\Phi} - \underline{\Phi}^T\underline{K}_R'\underline{B}\underline{G}_c^* - \underline{S}\underline{G}_c^* \qquad \text{(A-35a)}$$

$$= \underline{X} + \underline{\Phi}^T\underline{K}_R'\underline{\Phi} - [\underline{B}^T\underline{K}_R'\underline{\Phi} - \underline{S}^T]^T\underline{G}_c^* \qquad \text{(A-35b)}$$

which is the result we were seeking. So $\underline{K}_c$ and $\underline{K}_R'$ satisfy the same terminal condition and the same difference equation, so they are equal for all time.

Appendix B

CGTPIQ User's Guide

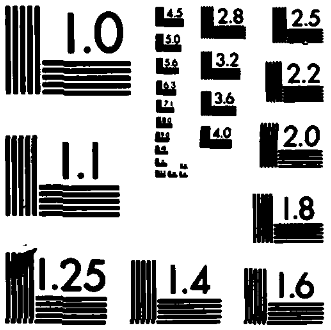
B.1 Introduction

The primary objective of this thesis effort is to create a computer program with which to design CGT/PI/KF controllers with the PI controller based on the integral of the regulation error. This user's guide discusses the program which has been developed--hereafter to be referred to as CGTPIQ--as an existing program (as it executes under CYBER INTERCOM) and with the intention of providing information appropriate to successful execution when applied to the user's design problem. It discusses program operation from the input/output perspective: the specific input and output of each design/evaluation path and the terminology employed in each input/output item. It also discusses what the user must do both before and immediately following program execution. The discussion presented in this guide is based on the program developed using CGT/PI formulation 1 (see Section 3.3.2). Appendix C gives the required code changes for the other formulations, but they are not discussed here. The changes to program execution by each CGT/PI formulation is either very minor or non-existent. (Note: the reader will see duplicated material from other chapters. This is done so that this User's Guide will be complete by itself.)

While the specific test application for the program

AD-A138 453 COMMAND GENERATOR TRACKER SYNTHESIS METHODS USING AN LQG (LINEAR SYSTEM M..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. J P MCMILLIAN 3/4

UNCLASSIFIED DEC 83 AFIT/GE/EE/83D-46 F/G 9/2 NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1

Appendix B

CGTPIQ User's Guide

B.1 Introduction

The primary objective of this thesis effort is to create a computer program with which to design CGT/PI/KF controllers with the PI controller based on the integral of the regulation error. This user's guide discusses the program which has been developed--hereafter to be referred to as CGTPIQ--as an existing program (as it executes under CYBER INTERCOM) and with the intention of providing information appropriate to successful execution when applied to the user's design problem. It discusses program operation from the input/output perspective: the specific input and output of each design/evaluation path and the terminology employed in each input/output item. It also discusses what the user must do both before and immediately following program execution. The discussion presented in this guide is based on the program developed using CGT/PI formulation 1 (see Section 3.3.2). Appendix C gives the required code changes for the other formulations, but they are not discussed here. The changes to program execution by each CGT/PI formulation is either very minor or non-existent. (Note: the reader will see duplicated material from other chapters. This is done so that this User's Guide will be complete by itself.)

While the specific test application for the program in the context of this thesis has been related to aircraft control design, CGTPIQ is written to be applicable to a wide variety of control design problems. It has the following attributes:

1. CGTPIQ executes interactively
2. The program utilizes efficient array allocation
   a. Initial memory allocation easily set
   b. Dynamic array allocation within total memory allocated
3. Various modes of entry are possible for the dynamics models
4. Design paths are automatically followed, with user prompts at necessary decision points
5. Requests for input include informative prompts
6. Copious program output is provided
   a. Output most relevant to design decisions are provided directly to the terminal
   b. Additional detailed output is provided to a separate output file
7. Information relevant to design iteration is preserved
8. Information needed for evaluation by computer program "PFEVAL" is preserved (Ref 21)
9. Error checking is performed, and messages given as appropriate

CGTPIQ employs computational routines available in a library of matrix computer routines described in Ref (12). Exclusive of the library routines, the program has a length of more than 2900 lines of source code. The programming language employed is ANSI Standard FORTRAN IV. Although the resulting source code is highly portable, local memory utilization limits for interactive execution may impose constraints. In use on a Control Data Corporation CYBER machine, the necessary load size was achievable with no impact on the source code. Thus the existing source code is in a pure form for whatever system. The final program size is much greater than the normal 65000 octal word limitation of the CYBER interactive system. Thus, in order to achieve interactive operation and provide sufficient free memory for array allocation so that problems of large and variable dimensions can be treated, a CYBER Loader option referred to as "Segmentation" (Ref 7) must be used. (Note: even with segmentation the program requires nearly 67000 octal words.) The required segmentation code will be presented later in this guide.

CGTPIQ was written specifically to run interactively. Requests for input, while intentionally brief, tell the user what is expected of him--what, how many, and in what units, as appropriate. Output of information, relevant to design decisions, to the terminal is compact and automatically provided. Also, the user can determine the amount and category of output to the terminal in some cases, and according to need.

An objective of this thesis effort was for CGTPIQ to be similar to the program CGTPIF by Floyd (Ref 9), as modified by Moseley (Ref 21), and every effort was made for this to be the case. For this reason the two program's are almost identical in operation. Appendix A, which discussed the actual programming text changes, assumed that the user must have copies of Floyd's and Moseley's Theses to effect any change to the program. Appendix A was not discussed in detail since most users will not need to change the program. On the other hand, a complete user's guide is necessary for the user to be able to use the program; therefore, this user's guide is taken entirely from the theses by Floyd and Moseley (Refs 9 and 21), with necesary changes to reflect the changes made in the program, so that the user will not need copies of the other two theses to use this program.

B.2 Program Operating Principles and Organization

CGTPIQ has three design paths: (1) design of a PI controller (using explicit and/or implicit quadratic weights); (2) design of either an open-loop CGT or closed-loop CGT/PI controller, and (3) design of a Kalman filter. Corresponding to the first two design options is a controller evaluation set of routines, and corresponding to the third design option is a set of routines for filter evaluation. The evaluation routines perform the computations discussed in Appendix D.

A general flowchart of CGTPIQ is given in Figure B-1, showing the main execution paths and design entries. The controller sample period is entered, the design model is established, and then the desired design path can be

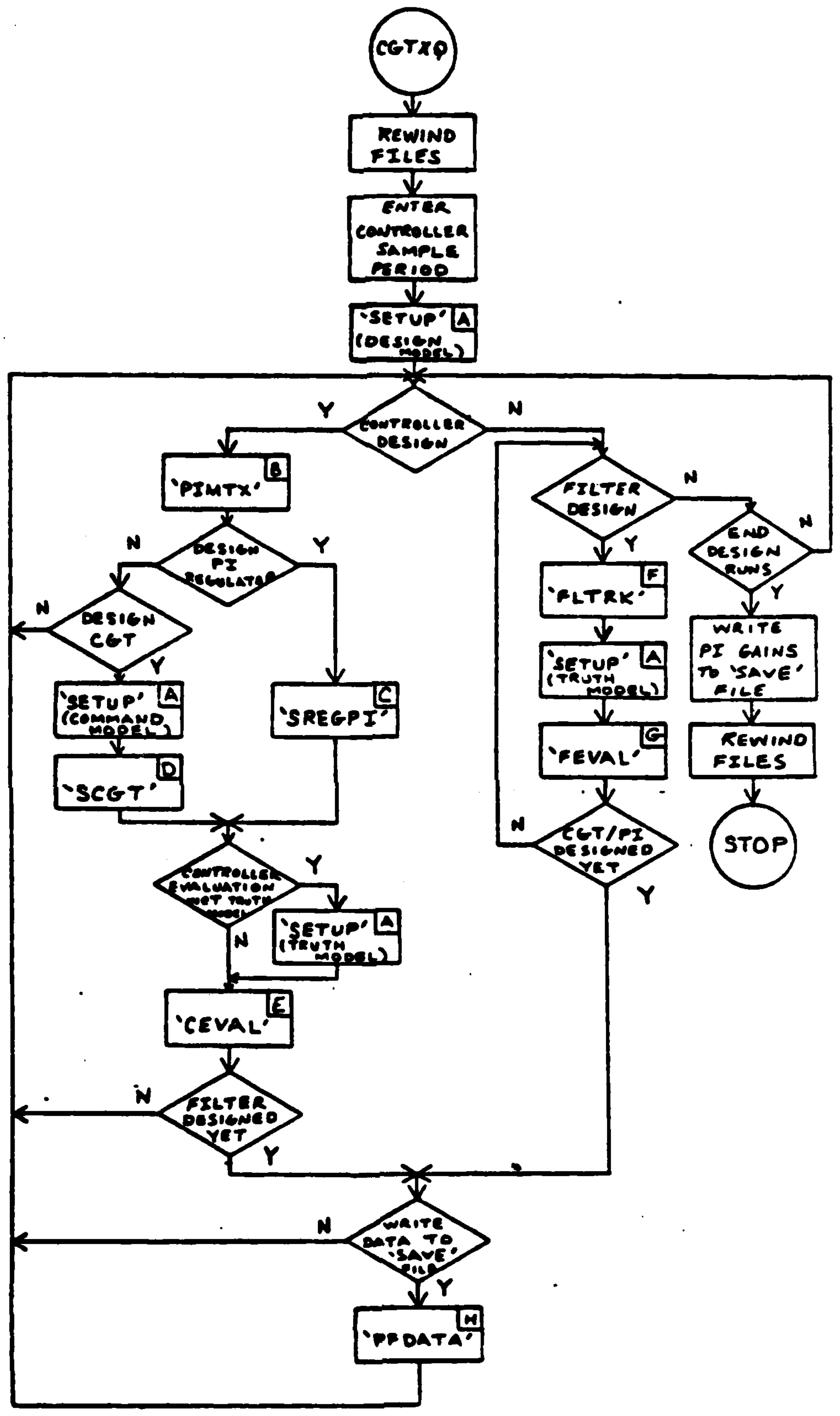


Fig. B-1. CGTPIQ General Flowchart

followed. The elementary design path choice is between controller and filter designs. The CGT or CGT/PI, and the PI controller (for explicit and/or implicit quadratic weights) design paths are then options within the controller design. If the PI controller design is pursued prior to the CGT design path, the CGT design will automatically be of the CGT/PI controller. If the PI is not already determined during the current execution of the program, then the designer may elect ot design either a CGT or a CGT/PI controller. However, the CGT controller design is not pursued if the open-loop design model is unstable. The controller design path is followed automatically by the appropriate controller evaluation path. Similarly, the filter design path leads automatically to the filter evaluation. When the evaluation is complete the designer is given the opportunity to loop on the design path, choose a different design path, or terminate program execution. More detailed flowcharts and functional diagrams are given later.

B.3 Dynamics Models

Each of the dynamics models entered into CGTPIQ is represented by a set of continuous-time state differential equations. The names used in the equations to follow are exactly those used by CGTPIQ in reference to these same dimensions and arrays in its Input/Output (I/O). Note that here each name is a single character, possibly subscripted, while in its I/O, CGTPIQ incorporates subscripts into the name (e.g., $\underline{A}_t$ becomes "AT"). Constraints on the models that are presented below are tested by CGTPIQ and, if not satisfied, a message is written to the user terminal and execution is aborted. These will be discussed later.

B.3.1 Design Model

The design model is given by

$$\dot{\underline{x}}(t) = \underline{A}\underline{x}(t) + \underline{B}\underline{u}(t) + \underline{E}_x\underline{n}_d(t) + \underline{G}\underline{w}(t) \tag{B-1a}$$

$$\dot{\underline{n}}_d(t) = \underline{A}_n\underline{n}_d(t) + \underline{G}_n\underline{w}_d(t) \tag{B-1b}$$

$$\underline{y}(t) = \underline{C}\underline{x}(t) + \underline{D}_y\underline{u}(t) + \underline{E}_y\underline{n}_d(t) \tag{B-1c}$$

$$\underline{z}(t_i) = \underline{H}\underline{x}(t_i) + \underline{H}_n\underline{n}_d(t_i) + \underline{v}(t_i) \tag{B-1d}$$

and $\underline{w}$, $\underline{w}_d$ and $\underline{v}$ are independent zero-mean noises with statistics

$$E\{\underline{w}(t)\underline{w}^T(t+\tau)\} = \underline{Q}\delta(\tau) \tag{B-2a}$$

$$E\{\underline{w}_d(t)\underline{w}_d^T(t+\tau)\} = \underline{Q}_n\delta(\tau) \tag{B-2b}$$

$$E\{\underline{v}(t_i)\underline{v}^T(t_j)\} = \underline{R}\delta_{ij} \tag{B-2c}$$

In these equations, $\underline{x}$, $\underline{u}$, $\underline{n}_d$, $\underline{y}$, and $\underline{z}$ are the system state, input, disturbance state, output, and measurement vectors, respectively. In input prompts, CGTPIQ refers to the diagonal elements of the noise covariance matrices as

$\underline{Q}$: "state noise strengths" (B-3a)

$\underline{Q}_n$:"disturbance noise strengths" (B-3b)

$\underline{R}$: "measurement noise strengths" (B-3c)

Note the $\underline{Q}$, $\underline{Q}_n$, and $\underline{R}$ are all assumed to be diagonal matrices. The dimensions of the model are

n = number of system states

r = number of system inputs

p = number of system outputs

m = number of system measurements

d = number of disturbance states

w = number of independent system noises

w_D = number of independent disturbance noises (B-4)

and the dimensions of the matrices of the model are

$\underline{A}$:	n-by-n
$\underline{B}$:	n-by-r
$\underline{E}_x$:	n-by-d
$\underline{G}$:	n-by-w
$\underline{Q}$:	w-by-w
$\underline{C}$:	p-by-n
$\underline{D}_y$:	p-by-r
$\underline{E}_y$:	p-by-d
$\underline{H}$:	m-by-n
$\underline{H}_n$:	m-by-d
$\underline{R}$:	m-by-m
$\underline{A}_n$:	d-by-d
$\underline{G}_n$:	d-by-w_D
$\underline{Q}_n$:	w_D-by-w_D

(B-5)

CGTPIQ requires that the numbers of design systems inputs and outputs be equal: r=p. Also, the number of system states may not be less than the number of disturbance states, due to the computational setup used for the CGT solution. The dimensions n, r, and p must be non-zero; any of the other dimensions may be zero. If m is zero or if w and w_D are both zero, the Kalman filter design path cannot be pursued. Matrices having either dimension zero, do not exist and are not entered.

B.3.2 Truth Model

The truth model is specified by

$$\dot{\underline{x}}_t(t) = \underline{A}_t\underline{x}_t(t) + \underline{B}_t\underline{u}_t(t) + \underline{G}_t\underline{w}_t(t) \qquad \text{(B-6a)}$$

$$\underline{z}_t(t) = \underline{H}_t\underline{x}_t(t_i) + \underline{v}_t(t_i) \qquad \text{(B-6b)}$$

$$\underline{x}'(t) = \underline{T}_{DT}\underline{x}_t(t) \qquad \text{(B-6c)}$$

$$\underline{n}_d'(t) = \underline{T}_{NT}\underline{x}_t(t) \qquad \text{(B-6d)}$$

and $\underline{w}_t$ and $\underline{v}_t$ are assumed to be independent zero mean noises with

$$E\{\underline{w}_t(t)\underline{w}_t^T(t+\tau)\} = \underline{Q}_t\delta(\tau) \qquad \text{(B-7a)}$$

$$E\{\underline{v}_t(t_i)\underline{v}_t^T(t_j)\} = \underline{R}_t\delta_{ij} \qquad \text{(B-7b)}$$

where Equations (B-6c) and (B-6d) relate the system and disturbance states of the design model to the system states of the truth model. Note that $\underline{Q}_t$ and $\underline{R}_t$ are both assumed to be diagonal matrices. In these equations, $\underline{x}_t$, $\underline{u}_t$, and $\underline{z}_t$ are the truth model system state, input, and measurement vectors, respectively. The vectors $\underline{x}'$ and $\underline{n}_d'$ are as defined for the design model.

The dimensions of the truth model are

n_T = "number of system states"

r_T = "number of system inputs"

m_T = "number of system measurements"

w_T = "number of independent noises driving system dynamics" (B-8)

and the dimensions of the matrices of the model are

$\underline{A}_t$: n_T-by-n_T

$\underline{B}_t$: n_T-by-r_T

$\underline{G}_t$: n_T-by-w_T

$\underline{Q}_t$: w_T-by-w_T

$\underline{H}_t$: m_T-by-n_T

$\underline{R}_t$: m_T-by-m_T

$\underline{T}_{DT}$: n-by-n_T

$\underline{T}_{NT}$: d-by-n_T (B-9)

CGTPIQ requires that the numbers of inputs and of measurements for the truth and design model be the same: r_T=r and m_T=m. If the number of driving noises (w_T) is zero, evaluation of a Kalman filter design is not pursued (since a covariance analysis with no truth model driving noise would not be very informative). Matrices having either dimension zero, do not exist and are not entered.

B.3.3 Command Model

The command model is given by

$$\dot{\underline{x}}_m(t) = \underline{A}_m\underline{x}_m(t) + \underline{B}_m\underline{u}_m(t) \tag{B-10a}$$

$$\underline{y}_m(t) = \underline{C}_m\underline{x}_m(t) + \underline{D}_m\underline{u}_m(t) \tag{B-10b}$$

In these equations, $\underline{x}_m$, $\underline{u}_m$, and $\underline{y}_m$ are the command model state, input, and output vectors, respectively.

The dimensions of the command model are

n_M = "number of model states"

r_M = "number of model inputs"

p_M = "number of model outputs" (B-11)

and the dimensions of the matrices are

$\underline{A}_m$: n_M-by-n_M

$\underline{B}_m$: n_M-by-r_M

$\underline{C}_m$: p_M-by-n_M

$$\underline{D}_m: \quad p_M\text{-by-}r_M \tag{B-12}$$

CGTPIQ requires that the number of outputs of the command and design models are equal: $p_M = p$. Also, the number of system states of the command model (n_M) cannot be greater that the number of system states of the design model (n). This constraint is due to the setup for computation of the CGT solution. (Note: the command model for explicit and implicit model following need not be the same, though they may be.)

B.3.4 Overall Model Dimensions

Problems of the following dimension can be accommodated by CGTPIQ

$$\begin{aligned}
n &\le 15\\
n+d &\le 15\\
r &\le 5\\
p &\le 5\\
m &\le 15\\
w &\le 15\\
w+w_D &\le 15\\
n_M &\le 10\\
r_M &\le 5\\
p_M &\le 5\\
n_t &\le 20\\
r_t &\le 5\\
m_t &\le 15\\
w_t &\le 20
\end{aligned} \tag{B-13}$$

NOTE, other combinations of dimensions, some greater than and some less than these specific dimensions, will also be accommodated due to the method by which arrays are allocated and stored. If the user has a problem that does not fit these size restraints he should try the problem to see if it will work. Messages will be printed if he has exceeded total allocation, and in that case, the basic program code will have to be altered to proceed.

B.4 Entry of Dynamics Models

Any of the three dynamics models described above may be established in any of three ways: (1) the dimensions and matrix elements may be entered directly from the terminal with input prompting from CGTPIQ; (2) the dimensions and matrix elements may be entered from a "DATA" file on which the dynamics model from a previous run of the program was written (the writing of such a file entry for each model is a program option); or (3) the dimensions and matrix elements may be established by user provided subroutines as was done for some of the airplane models used and discussed in Floyd's thesis (Ref 9). These models are also in CGTPIQ.

When entered from the terminal directly or set by subroutines, only non-zero elements of the various matrices need be given. In many cases, this substantially simplifies establishing the matrices and reduces the probability of erroneous entries. Subsequent design runs for the same problem can then simply read the models from the "DATA" file previously created.

Each request for input from the user includes a prompt by CGTPIQ of information defining the input desired, whether this input is the dynamics models or any other required input. When an option is offered, the prompt succinctly describes the option and specifies the appropriate form of response (e.g.,"Y or N", signifying and yes or no response.) When a dimension is to be entered, the dimension is identified. When a matrix is to be entered, the matrix is identified, as is its dimension, and the appropriate form of input. Similarly, all other requests for input identify the input expected and the form the entry is to take. The messages to the user are brief and require that the user have some understanding of what is involved in achieving each of the designs. The prompts are intended to assist users familiar with the elements of the PI, CGT, CGT/PI, and KF design methods and with the terminology used in this thesis to enter the necessary information for such designs into the program.

B.5 Program Output

In the computations involved in the various design paths, a great deal of information is generated. While all the information generated may be relevant to the design, in the usual case only a small fraction of the information is needed to pursue iteration of the design paths. Information most relevant to achieving the various designs is output to the terminal either automatically or at the user's option. The same information, along with all other potentially useful information generated by the program, is also output to a "LIST" file. After program execution is complete, the user may then look through this file at the terminal or may "ROUTE" it to a line printer for a complete listing. All output is given an identifying name or description which is consistent with the terminology of this thesis.

Plots are available as options of the design evaluation routines. The plots produced are of the "line printer" type. For the controller, plots of selected variables may be output to the terminal, in addition to the full set of plots output to the "LIST" file.

During program execution, user entries in the design iteration are preserved. Thus, for each iteration only those entries to be changed need be given as input, making design iteration both fast and easy in terms of the simple mechanics of the process. Also, computations which are not modified by design iteration are performed only once and the results are preserved for reuse within the current run, as needed.

B.5.1 File Usage: PLOT, SAVE, DATA, LIST

In addition to the I/O communication directly with the user terminal, CGTPIQ employs four disk files for I/O purposes. One of these ('PLOT') is for temporary use by CGTPIQ only. The other three files ('SAVE', 'DATA', and 'LIST') benefit the user by providing continuity between distinct executions of the program (SAVE, DATA) or providing supplementary design output data (LIST).

CGTPIQ preserves information for use in distinct execution of the program through use of the SAVE and DATA files. During program execution, the dynamics models as well as the PI controller gains (if available) may be written to the SAVE file. Following execution, the user may wish to catalog SAVE as a permanent file. In subsequent executions, CGTPIQ may (at the user's option) read any of the dynamics models or PI gains from the file named DATA. (Also the gains to be used by the program "PFEVAL" are also written to the SAVE file). Both files are rewound prior to and following program execution. Letting the abbreviations "BE" and "AE" mean before and after execution, respectively, typical operations on these files include:

1. Catalog SAVE file:
   a. BE:REQUEST,SAVE,*PF
      AE:CATALOG,SAVE,pfn
   b. AE:REQUEST,DUM,*PF
      AE:COPYBF,SAVE,DUM
      AE:CATALOG,DUM,pfn
2. Attach DATA file:
   BE:ATTACH,DATA,pfn
3. Reuse SAVE file as new DATA file
   AE:RETURN,DATA
   BE:COPYBF,SAVE,DATA

None of these operations are required; they are simply useful operations in the event the user chooses to employ the files to streamline repeated executions of a given design problem. Note that SAVE and DATA are local file names and that the permanent file names are represented here by the abbreviations "pfn". Other operations (e.g., PURGE) and other combinations of operations are possible as for any files, and the usual rules for these operations apply here as well. The essential points to understand are that the SAVE file is created by CGTPIQ and is an output file only, and that DATA is a previously SAVE'd file under a new local file name and is an input file only. During a single program execution the two files are distinct and these roles cannot be changed.

During program execution, extensive design information is output to the LIST file. After execution is complete, the user may wish to route LIST to a line printer for listing (or it may be "PAGED" at the user terminal). The file is rewound before and after execution. To send LIST to a line printer, the following command is used after execution:

```
ROUTE,LIST,DC=PR,TID=nn,ST=CSB,FID=abc
```

in which "nn" is the identification number of the terminal to which the file is to be sent (for AFIT ,nn=AF), and "abc" is any three character output banner for the listing.

The PLOT file is used internally by CGTPIQ for temporary storage of the variables generated by time response evaluations of the controller or filter. If desired, it may be eliminated following execution using the command:

```
RETURN,PLOT
```

B.6 Error Checking

A common problem encountered in executing computer programs with variable dimension array storage is the unintentional (and often unknown) over-running of the allocated storage area. When this happens, the program may fail (due to the over-writing of program code, for example) or, even worse, the program may appear to run properly but provide erroreous results. To avoid these difficulties, before using each of the Common arrays, CGTPIQ computes the allocation needed for the arrays it will generate and compares it with the number of words of memory actually allocated. If more memory is needed than has been provided, a message is written indicating the problem, the Common in question, and the minimum allocation needed. Execution is then aborted.

Error checks for array entry from the terminal are also performed. Not only are the array dimensions identified in prompts requesting entry of arrays, but each entry is checked to verify that it is within the row, column bounds for the array. If not within bounds, the entry is not accepted and a message identifying the problem is given. Also, for matrices requiring special properties (e.g., positive semi-definiteness), entries which clearly violate these requirements (e.g., negative diagonal elements) are not accepted and a message is written to the terminal.

Various other error checks are performed for each input entered from the terminal. These checks ensure that no obviously invalid entries are accepted (e.g., non-positive controller sample time), and no storage out of array bounds occurs. Obviously, it is not possible to guard against valid yet erroneous entries, but in most cases the program provides opportunities to correct mistaken entries made and discovered by the user. Additional tests are performed to ensure dimensional consistency of the dynamic models. These dimensions were discussed previously. For example, CGTPIQ checks that the numbers of inputs and outputs defined for the design model are equal, as well as checking each of the other dimensional conditions as each model is established. If the condition is not met, a message identifying the problem is written and the program execution is aborted.

B.7 Preparation Prior to Program Execution

Preparation for use of the program consists primarily in determining the dimensions of the models to be employed and the specific coefficients of the matrices comprising those models. It is appropriate for the PI design that an initial set of quadratic weights be established also, in order to begin the design iteration process.

Assuming that the user understands the nature and requirements of the various design elements, and is familiar with terminology related to such designs, the actual execution of the program involves straightforward response to input prompts. The specific meaning of all program prompts employed will be discussed later. Information most useful to the design iterations is available directly as output to the terminal, while additional information is provided in the separate output 'LIST' file. The terminology employed in the output is defined later.

B.7.1 Determine Dynamics Models

CGTPIQ employs three dynamics models as discussed previously. It is necessary that the user determine the dimensions and parameters of these models prior to execution of the program. The specific models needed to execute the design paths of interest need be known. At a minimum, the design model must be known in order to execute any of the designs. The truth model is required for evaluation of the Kalman filter (to perform a covariance analysis) and is optional for evaluation of the PI controller or CGT and CGT/PI controllers. The command model must be known in order to effect either CGT or CGT/PI designs. The dynamics models are entered into the program during execution as needed and under input prompting provided by CGTPIQ.

B.7.2 Define Objectives and Specifications

Before embarking upon design of the controller, the designer should define: (1) the objectives which are to be sought, and (2) appropriate specifications and constraints to apply to the controller. These may be rather loosely defined initially, then become more specific and firm as the design progresses. Objectives will vary with the problem under consideration, but might be exemplified by formulation of a desired controlled output response behavior. Specifications and constraints derive from the problem application and from the objectives for the design. Typical considerations include time-delay, overshoot, and settling time of the response, and input magnitudes and rate limits.

B.7.3 Determine Appropriate Initial Quadratic Weights

Execution of the PI controller design entails entry of quadratic weights for the optimal cost function. Such weighting matrices are required for the outputs, the input magnitudes, and the integral of the regulation error (also implicit weights to be discussed later). For these, only diagonal elements are required as input, since CGTPIQ assumes them to be diagonal matrices. However, after CGTPIQ computes the resulting augmented state and integral of the regulation error weighting matrix (Equation (III-36c)), the user may modify any element of it to achieve design goals.

Although final selection of appropriate quadratic weighting values to achieve design requirements is achieved in an iterative (trial-and-error, hopefully with insight) fashion, it is possible to make initial choices which are plausible. A common method for determining initial quadratic weights involves inverse square weighting of maximum deviations of outputs and inputs to achieve regulation for an assumed perturbation of the system (Refs 15 and 18:10-11). For example, the diagonal output weighting matrix element $\underline{Y}_{ii}$ would be

$\underline{Y}_{ii} = 1./(\text{maximum allowable } \underline{y}_i)^2$.

Beginning with the initial set of quadratic weights, the PI design path is executed repeatedly with changes in the choice of weighting elements until the design is satisfactory. CGTPIQ provides information during execution which allows the design to be evaluated and iteration of the design to be pursued effectively.

For open-loop CGT designs and Kalman filter designs, preparation consists of defining the various dynamics models. The open-loop CGT design depends only on the design and command models. The initial execution of the Kalman filter path depends only on the design model (and truth model for evaluation). Further refinements to either design are achieved through modifications of the appropriate dynamics model (command or design).

B.8 CGTPIQ Execution

B.8.1 Introduction

An important feature of CGTPIQ is that it follows appropriate paths through execution automatically, prompting the user for input as necessary. The basic design paths are selected by the user under prompting, but within a given path, only information needed to execute the specific design and evaluation is requested by the program. The user thus does not need any predetermined sequence of command entries to the program, nor are the commands coded in any way.

Figure B-1 gave a general flowchart of CGTPIQ. Each of the decision blocks (diamond shaped) represent a prompted request for input to choose the design to be pursued. Each rectangular block with an alphabetic character ("A" through "H") in the lower right corner represents a computational element of CGTPIQ and is discussed individually in the following subsections. The subsections which follow discuss the I/O of each computational element. In identifying items of I/O, reference will sometimes be made to array names and equations which are specified elsewhere in this thesis (equation references will be in parentheses following the array name). All prompts for input define the input that is being requested and the manner in which the entry should be given. Since the actual prompts are themselves understandable, they will not be quoted here. Instead, flowcharts will be used to show where prompts occur and how execution depends on user entries. Blocks involving I/O will be identified by function: an "I" block will signify prompted input from the terminal; an "OT" or "OL" block will signify output to the user terminal or LIST file, respectively. All output to LIST is separated and identified according to the computational element which generated it.

B.8.2 Types of Entries

Required inputs may entail entry of a decision logic value, a single numerical or character value, or multiple numerical values for arrays or vectors. In all cases, CGTPIQ prompts the user with messages identifying the nature of the input requested and each prompt ends with the character ">".

All requests for decisions affecting execution are framed as questions requiring a YES ("Y") or NO ("N") response. The user entry is read as character input. Execution proceeds according to a default "YES" assumption: all decision tests assume that if the answer is not "NO", then it is "YES".

Requests requiring single entry responses always specify the variable requested. If there are constraints on acceptable input they are indicated in the prompt and adherence is tested in the program after entry. If the entry is not "valid", a message is written to the terminal and the prompt is repeated.

All requests for entry of vector or array elements specify the name of the array in question and its actual dimensions. Entries for vector elements include an integer specifying the index of the element, and a real specifying its value. For most arrays, all elements may be given values, while for some square matrices, only diagonal elements may be set (the prompt format will distinquish between the two). In the usual case, elements are entered into arrays by specifying two integers for the [row,column] address and a real for the value of that element. In cases in which only diagonal elements can be specified, entry is the same as for vectors, with the matrix diagonal considered a vector. As many entries as desired may be made and any entry can be repeated (e.g., to correct previous erroneous entries). Entry is terminated by specifying a row index of zero. Each entry is tested to verify that it lies within the [row,column] bounds of the array (vector). If an index is not "valid", a message is written to the terminal indicating the error and the initial prompt with the array dimensions is given again (previous valid entries are not affected, only the specific invalid entry is rejected). If an entry is valid, the element value is set and the next entry is awaited without additional prompting. For example, if it is desired to set a square matrix of dimension three to an identity matrix, then according to whether the specific matrix is to be entered in [row,column] or diagonal form, entries would be as follows:

1. For [row,column] entry format

```
1 1 1.   (enter)
2 2 1.   (enter)
3 3 1.   (enter)
0/
```

2. For diagonal element entry format

```
1 1.     (enter)
2 1.     (enter)
3 1.     (enter)
0/
```

Items of information may be separated by one or more blanks or by a comma. These entries set specific elements of the matrix to non-zero values (the matrix was initialized automatically with all elements zeroed). The following sections will now discuss the individual boxes denoted with a letter in Figure B-1.

B.8.3 Establishing Dynamics Models ("A")

All three dynamics models (design, truth, and command) are entered under the control of a single set of routines. The options for entry and the type of I/O involved for each is of identical format, but prompts and output employ terminology specific to each model to identify items of I/O.

Figures B-2a,b,c give flowcharts of the I/O involved in entry of the models. Note that any of the dynamics models may be entered in any of the following ways:

1. The dimensions and array elements may be read from the DATA file.
2. The dimensions and array elements may be entered from the user termnal as prompted by CGTPIQ.
3. The dimensions and array elements may be determined by user-provided subroutines.

These modes of entry are offered by CGTPIQ in the order above, with option 3 assumed selected if options 1 and 2 are declined. If option 1 is selected, the reading of the model is automatically performed. If the model is found not to exist in the DATA file, the other options are offered. For option 3, if the subroutines needed to define the model are not loaded, option 1 and 2 are offered again. This logic is illustrated in Figure B-2a.

Prior to entry of the model matrices, all matrix elements are initialized to zero. Using option 1, all array elements are then read automatically from the DATA file. For options 2 and 3 only the non-zero array elements must be

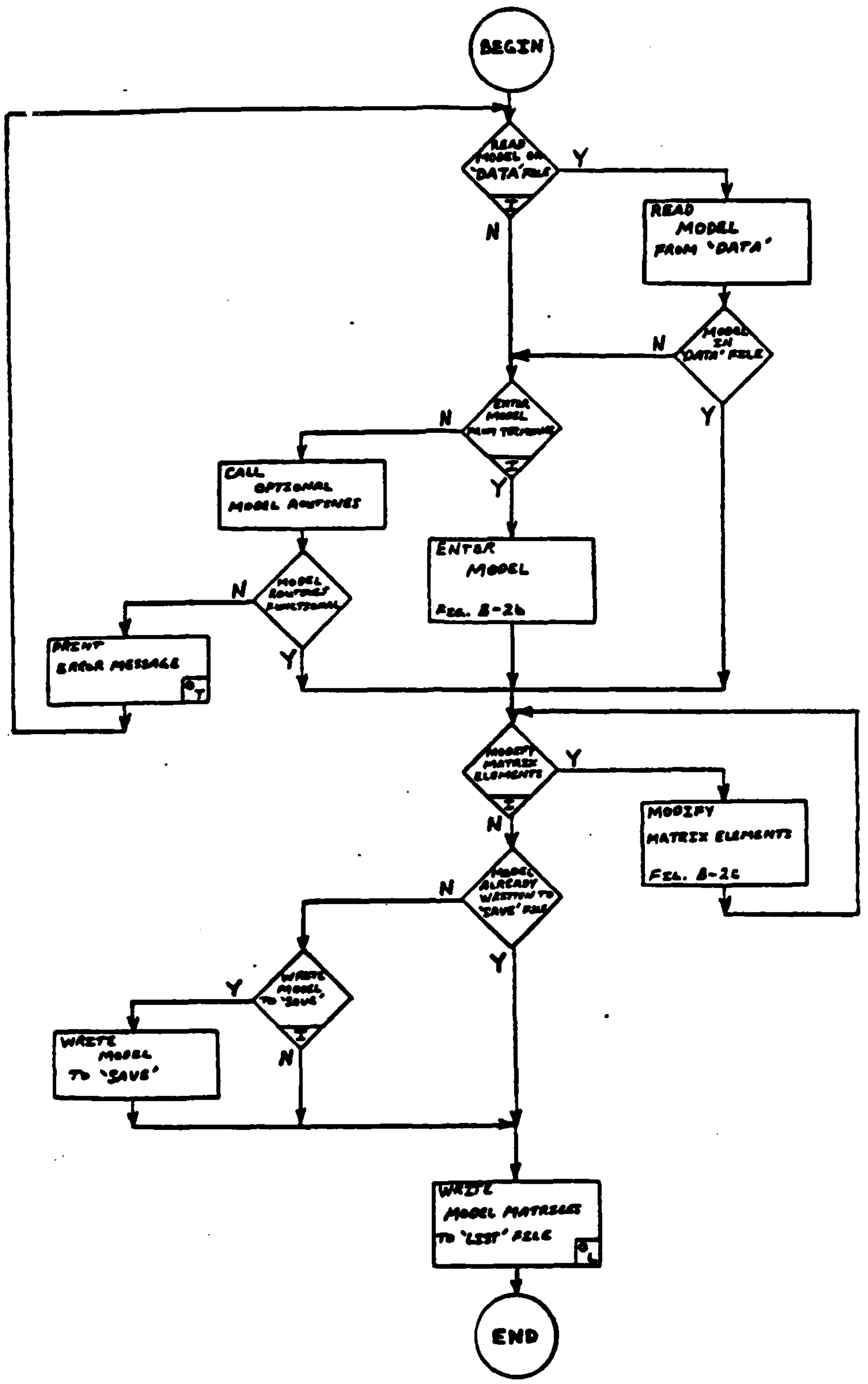

BEGIN
READ MODEL ON 'DATA' FILE
I
Y
N
READ MODEL FROM 'DATA'
MODEL IN 'DATA' FILE
N
Y
ENTER MODEL FROM TERMINAL
I
N
Y
CALL OPTIONAL MODEL ROUTINES
ENTER MODEL FIG. B-2b
MODEL ROUTINES FUNCTIONAL
N
Y
PRINT ERROR MESSAGE
T
MODIFY MATRIX ELEMENTS
I
Y
N
MODIFY MATRIX ELEMENTS FIG. B-2c
MODEL ALREADY WRITTEN TO 'SAVE' FILE
N
Y
WRITE MODEL TO 'SAVE'
I
Y
N
WRITE MODEL TO 'SAVE'
WRITE MODEL MATRICES TO 'LIST' FILE
L
END

Fig. B-2a. Dynamics Model Entry (Executive)

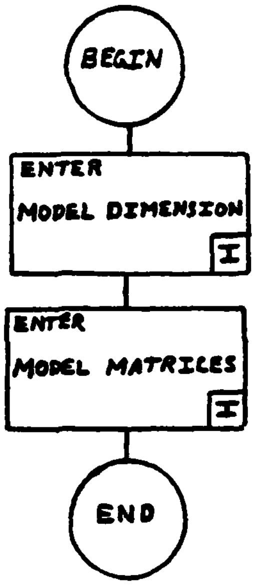


Fig. B-2b. Enter Dynamics Model from Terminal

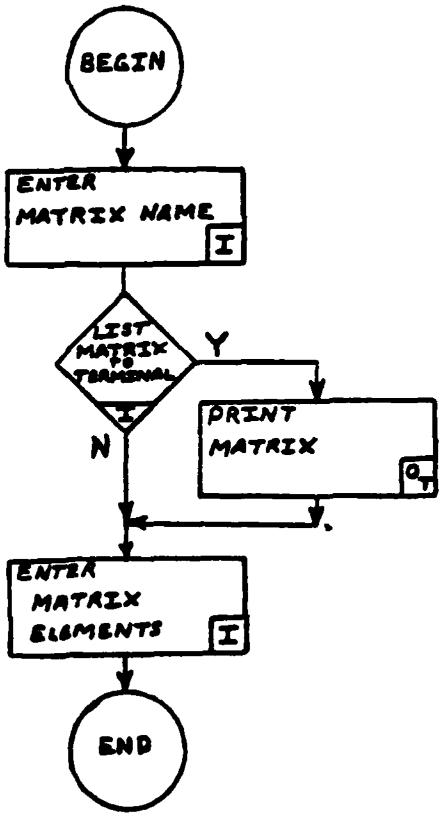


Fig. B-2c. Modify/List Model Arrays

established.

Figure B-2b illustrates model definition under option 2. Entry of the diminsions and arrays is according to prompts by CGTPIQ. The dimensions are requested first by the names and in the order of Equation (B-4), (B-8), or (B-11), as appropriate. The arrays are then requested, also by name and in the order of Equation (B-5), (B-9), or (B-12), as appropriate. An array is not requested if its dimension is zero. Each prompt includes the actual dimensions of the array according to the model dimensions previously entered. Elements of arrays are entered by address, and value by giving the [row,column] address and element value as a three item input. Entry of a zero row address terminates entry of the array.

For option 3, each model requires two user-provided routines of prescribed names, argument lists, and characteristics. These must be compiled with the main routine of CGTPIQ and a segmented executable object file created. For each model defined by subroutines, one subroutine must establish the dimensions of the model, and another must set the values for all matrices of that model. Each routine must have the appropriate name and argument list. All model arrays appearing in the argument lists must be allocated in full manner: the array dimensions specified by "Dimension" statements within the routines must be exactly those implied by the routine specifying model dimensionalities and the array sizing given by Equations (B-5), (B-9), and (B-12). For example, if the number of design model states (n) is established as 10, then according to Equation (B-5) the system matrix must be explicitly dimensioned A(10,10) in the subroutine which sets array values for the design model. All of these arrays are initialized to zero before the array setting routines are called, so it is necessary only to set non-zero array elements within the subroutines. Any arrays of dimension one in both row and column are actually scalars and need not be included in a Dimension statement. Any arrays with row or column dimension of zero are in fact nonexistent arrays and must not be included in Dimension statements, although they still must be included in the subroutine's argument list (since calls to these routines from within CGTPIQ assume full argument lists).

After a model is defined using any of the three entry options, the user may list any matrix and modify any array elements, again under prompting by CGTPIQ. If a modification/list is desired, the names of the model's matrices are listed at the terminal and the user specifies the array of interest by name. Elements are entered by address and value as described previously. Figure B-2c illustrates the I/O involved in modifying/listing model arrays.

When the model has been defined to the user's satisfaction, it may be written to the SAVE file by CGTPIQ if the user chooses. In the course of design iteration, the truth and command models may be redefined if desired, but only a single copy of any model may be written to the SAVE file during a given execution of the program (CGTPIQ will not offer additional opportunities after a given model has been SAVE'd).

For each model, the discrete-time representation is computed for the controller sample period specified. Later computations do not depend on the continuous-time dynamics models, so the arrays defining them are not retained. (Note: To consider effects of different sample periods, the program will require re-execution with re-entry of the models.)

Arrays defining both the continuous-time and discrete-time models are given in output to the LIST file. The specific output items, their names, and the reference equations are listed below for each model

<u>Design Model</u>

Continuous-time model matrices as listed in Equation (B-5), plus discretized model matrices as:

PHI:	$\underline{\Phi}$	(IV-5)
BD:	$\underline{B}_d$	(IV-5)
QD:	$\underline{Q}_{ad}$	(IV-6c)
HA:	$\underline{H}_a$	(IV-1d)
EXD:	$\underline{E}_{xd}$	(IV-7a)
PHN:	$\underline{\Phi}_n$	(IV-7b)

<u>Command Model</u>

Continuous-time model matrices as listed in Equation (B-12), plus discretized model matrices as:

PHM:	$\underline{\Phi}_m$	(IV-17)
BDM:	$\underline{B}_{md}$	(IV-17)
CM:	$\underline{C}_m$	(IV-18)
DM:	$\underline{D}_m$	(IV-18)

Truth Model

Continuous-time model matrices as listed in Equation (B-9), plus discretized model matrices as:

PHT:	$\underline{\Phi}_t$	(IV-12)
BDT:	$\underline{B}_{td}$	(IV-12)
QDT:	$\underline{Q}_{td}$	(IV-13c))

In addition, the eigenvalues of the system matrices of each model ($\underline{A}$, $\underline{A}_m$, $\underline{A}_t$) are computed and output both to the user terminal and the LIST file. The system model is identified by type (design, command, truth). Eigenvalues of the corresponding discretized system matrices are not computed.

B.8.4 Controller Setup ("B")

The "controller setup" routines perform computations needed for the controller designs. No input is required of the user and the output is to the LIST file only. The output is the matrix $\underline{\Pi}$:

PI:	$\underline{\Pi}$	(II-18)

B.8.5 PI Design ("C")

The PI design has as a first question to the user whether he wishes to use and/or combine implicit/explicit model following by incorporating implicit weights. If the user responds with no ("N") then execution of the PI design path entails user entry of only explicit quadratic weighting matrices defining the costs associated with output deviation, control magnitudes, and integrals of the regulation error (see Figure B-3)(the results of a yes answer to this question will be discussed later in this subsection):

OUTPUT DEVIATION:	$\underline{Y}$	(A-3a)
CONTROL MAGNITUDES:	$\underline{U}_y$	(A-5b)
INTEGRAL OF THE REGULATION ERROR:	$\underline{U}_q$	(A-3b)

For each of these matrices, only the diagonal elements are entered. On the first execution of the PI design, all weighting matrices are initialized to zero. Subsequent iterations preserve the elements of these matrices so only desired changes in specific weighting elements need be entered. Weights on output deviations should be non-negative, while weights on control magnitudes and integrals of the regulation error should be positive. Entries are tested for positive or non-negative values as appropriate. If an entry is not valid, a message is written to the user terminal and that entry is not accepted.

Matrices $\underline{Y}$ and $\underline{U}_q$ are used to compute a quadratic weighting matrix on the state deviations (using an augmented state vector composed of system state and integral of the regulation error perturbations from nominal). This

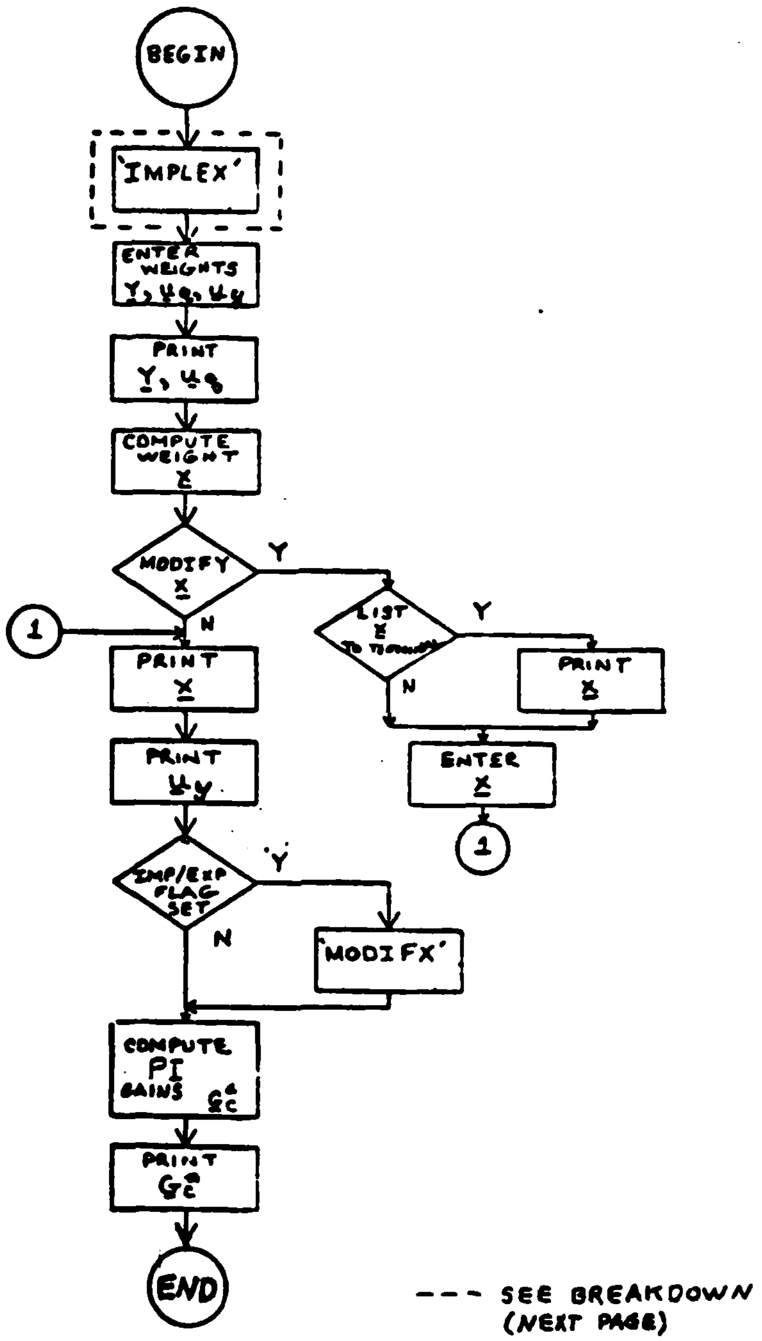
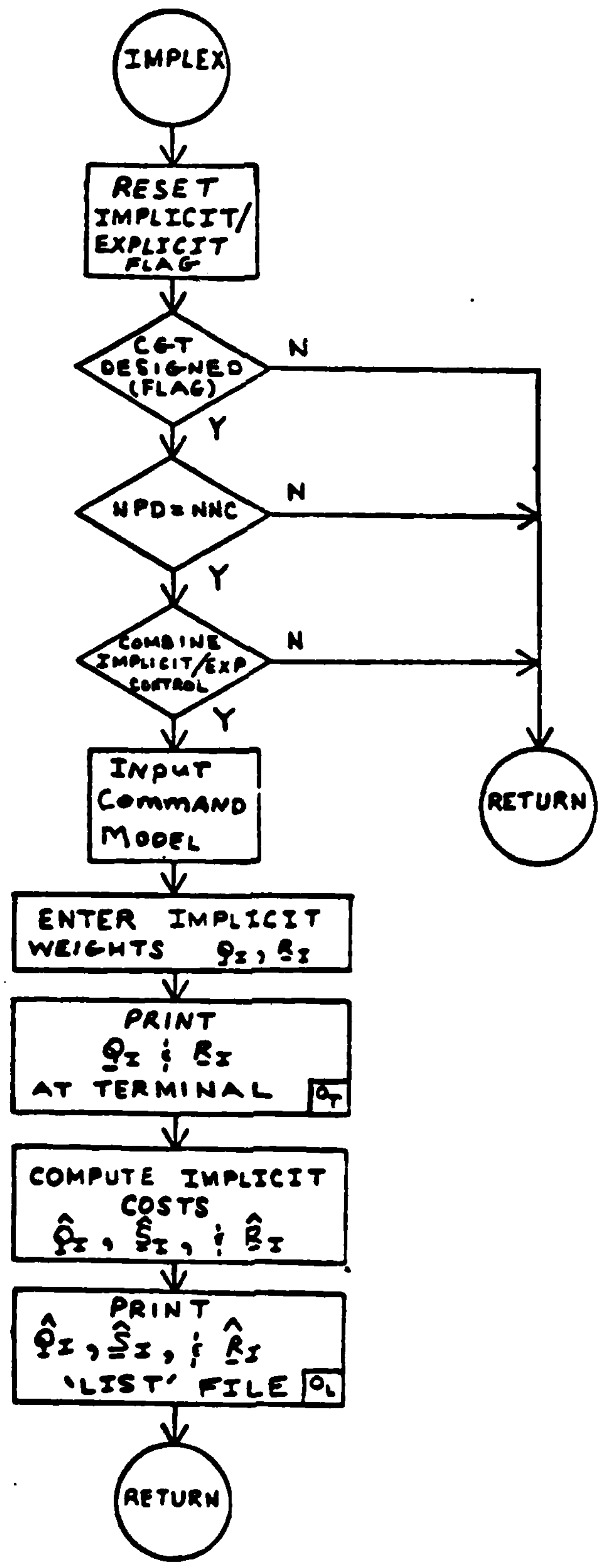


Fig. B-3. PI Controller Design



IMPLEX
RESET IMPLICIT/ EXPLICIT FLAG
CGT DESIGNED (FLAG)
N
Y
NPD = NNC
N
Y
COMBINE IMPLICIT/EXP CONTROL
N
Y
Input Command Model
RETURN
ENTER IMPLICIT WEIGHTS $\underline{Q}_I$, $\underline{R}_I$
PRINT $\underline{Q}_I$ & $\underline{R}_I$ AT TERMINAL
O_T
COMPUTE IMPLICIT COSTS $\hat{\underline{Q}}_I$, $\hat{\underline{S}}_I$, & $\hat{\underline{R}}_I$
PRINT $\hat{\underline{Q}}_I$, $\hat{\underline{S}}_I$, & $\hat{\underline{R}}_I$ 'LIST' FILE
O_L
RETURN

Fig. B-3-- Continued

new matrix is referred to as "$\underline{X}$" (Equations (III-34c) and (III-35a,b,c)). The user may then modify any element of $\underline{X}$ and/or list it at the terminal. Elements entered into $\underline{X}$ are automatically set symmetrically by CGTPIQ, but the sign of diagonal elements entered is not tested. $\underline{X}$ is not preserved between PI design iterations, so any desired changes in elements with respect to their values as determined from $\underline{Y}$ and $\underline{U}_q$ must be re-entered each design pass.

The diagonal elements of $\underline{Y}$, $\underline{U}_q$, and $\underline{U}_y$ are printed at the user terminal and the entire $\underline{Y}$, $\underline{U}_q$, $\underline{U}_y$, and $\underline{X}$ matrices are output to the LIST files. Next, the controller gains and PI gains are computed. The PI gains are printed at the terminal ("GC1" and "GC2") and all gains are output to the LIST file. The intermediate product "E" is also output to the LIST file. The proportional "KP" and integral "KI" gains of the PI controller are also output to the LIST file.

The outputs to the LIST file are:

Y:	$\underline{Y}$	(A-3a)
UQ:	$\underline{U}_q$	(A-3b)
UM:	$\underline{U}_y$	(A-5b)
E:	$\underline{E}$	(III-28a)
GC1:	$\underline{G}_{c1}^*$	(III-20)
GC2:	$\underline{G}_{c2}^*$	(III-20)
KI:	$\underline{K}_i$	(III-31c)
KP:	$\underline{K}_p$	(III-31b)

Note that the mnemonics "UQ" and "UM" refer to integral of the regulation error and input magnitude weighting matrices, respectively.

If the user responded with a yes to the first question of combining implicit/explicit control, the program prompts the user with messages identifying the required implicit model-following related inputs. Entries are requested for the command model first (this need not be the same command model as in explicit design but the number of model states must equal the number of system outputs, nM=p), and then for the quadratic weighting on the implicit output derivatives and then the implicit control magnitudes. Note that only diagonal elements can be specified using the format discussed previously. After entry of the implicit quadratic weights, the user is permitted to see the "XIE" matrix (the upper left partition of Equation (III-98)) that is generated and he is asked if new implicit weights are required to be entered. If yes, the program re-initializes the implicit matrices and asks for the implicit weights again. If no, the implicit weighting matrices $\underline{Q}_I$ and $\underline{R}_I$ are then printed at the terminal. Matrix $\underline{Q}_I$ is the implicit output derivatives matrix; matrix $\underline{R}_I$ is the implicit control magnitude matrix. $\underline{Q}_I$ and $\underline{R}_I$ are equivalent to $\underline{X}_I$ and $\underline{U}_I$ of Equation (III-93), respectively. Following this, the program returns back for the normal explicit weights to be input as discussed previously. The user may design an implicit only (with limitations), explicit only, or a combined implicit/explicit controller via his selection of quadratic weights. An implicit model-following controller is pursued if the explicit quadratic weights are all chosen as zero (limitation: experience has shown that UQ should not be zero or the Riccati equation solver typically will not converge to a solution in the maximum permitted iterations). On the other hand, an explicit model-following controller is pursued if the implicit quadratic weights are chosen as zero; or if the user responds with no ("N") to the first question. If the implicit or combined implicit/explicit model-following is chosen the program utilizes the appropriate combined quadratic weights according to Equation (III-98). This also results in additional outputs to the LIST file

QI:	$\underline{X}_I$	(III-91)
RI:	$\underline{U}_I$	(III-91)
QIH:	$\hat{\underline{X}}_I$	(III-93a)
SIH:	$\hat{\underline{S}}_I$	(III-93b)
RIH:	$\hat{\underline{U}}_I$	(III-93c)
XIE:	$\underline{X}$(modified)	(III-98)

When execution of the PI design computations is complete, the "controller evaluation" set of routines is automatically executed. These are discussed in a later subsection as a separate computational element.

B.8.6 CGT Design ("D")

Execution of the "CGT design" path requires that a command model be established. If desired, a new command model can be established during any iteration of the design. The model is actually entered using the routines described in Section B.8.3 above ("Establishing Dynamics Models").

If PI gains already exist in the program storage, then a closed-loop CGT/PI design is effected automatically. If not, the user may elect to have the program read the PI gains from the DATA file and design a closed-loop CGT/PI controller. However, if the user chooses not to have the gains read from DATA or if the gains are found not to exist on the DATA file, an open-loop CGT design is effected automatically (by setting PI gains to zero), but only if the open-loop system is stable. For either open-or closed-loop CGT design, the matrices $\underline{A}_{ij}$ (Equation (III-61)) are automatically output to the LIST file.

Figure B-4 illustrates the I/O, logic, and computations of the CGT design path. Details involved in entering the command model are given in Section B.8.3 and are indicated in this figure by a block titled "Establish Command Model". Note that since the continuous-time representation of the command model is not preserved, "modification" of the command model actually entails complete redefinition of it. In case the command model exists on the DATA file and only specific elements are to be changed, this can be accomplished readily by reading the model from DATA and then modifying individual arrays (as shown in Figure B-2c).

In establishing the command model, I/O is as described in Section B.8.3. Additional output to the LIST file is

All: $\underline{A}_{11}$ (III-61a)

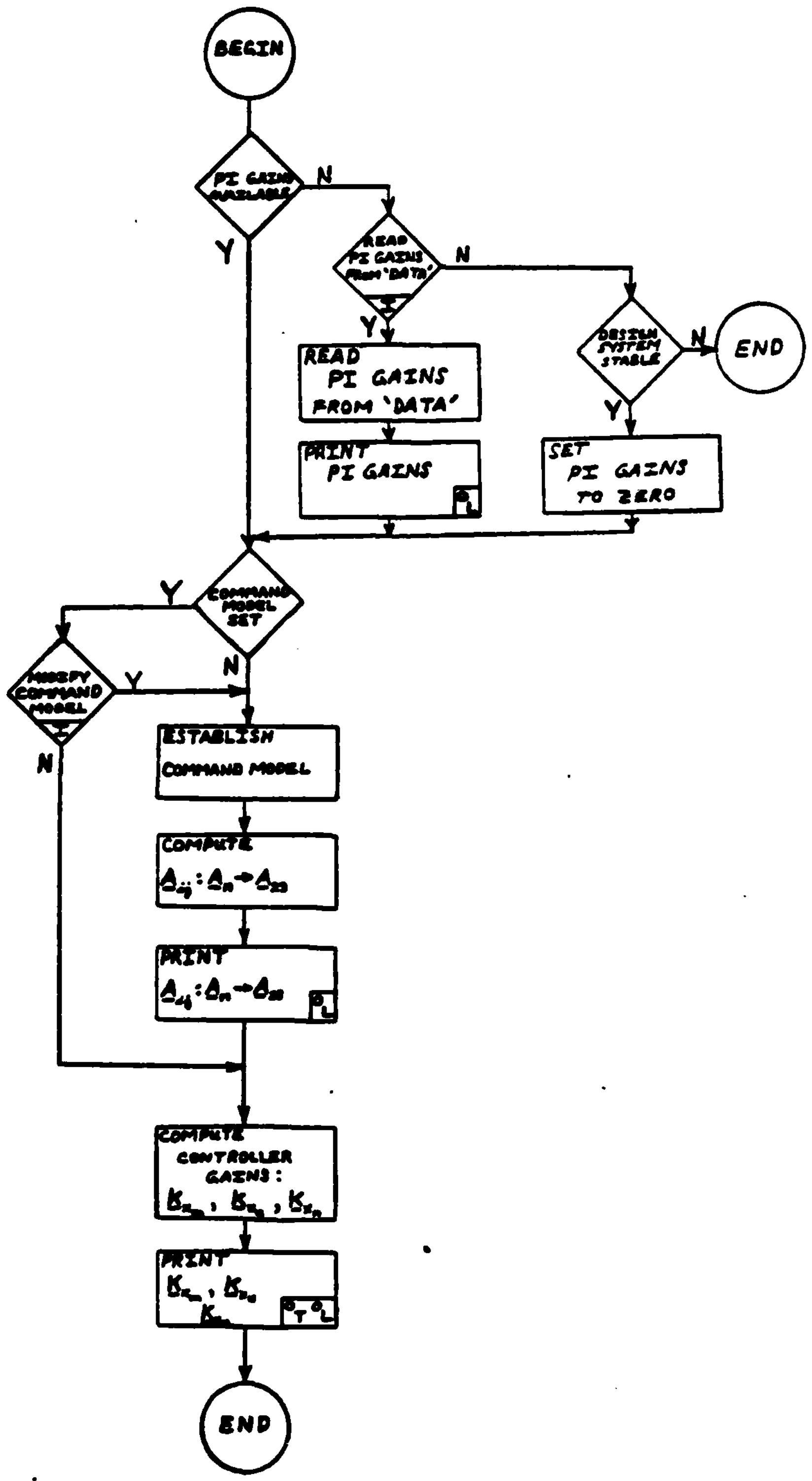

BEGIN
PI GAINS AVAILABLE
N
Y
READ PI GAINS FROM 'DATA'
N
Y
READ PI GAINS FROM 'DATA'
PRINT PI GAINS
DESIGN SYSTEM STABLE
N
END
Y
SET PI GAINS TO ZERO
COMMAND MODEL SET
Y
N
MODIFY COMMAND MODEL
Y
N
ESTABLISH COMMAND MODEL
COMPUTE $\underline{A}_{ij}:\underline{A}_{11}\rightarrow\underline{A}_{22}$
PRINT $\underline{A}_{ij}:\underline{A}_{11}\rightarrow\underline{A}_{22}$
COMPUTE CONTROLLER GAINS: $\underline{K}_{x_m}$, $\underline{K}_{x_u}$, $\underline{K}_{x_n}$
PRINT $\underline{K}_{x_m}$, $\underline{K}_{x_u}$ $\underline{K}_{x_n}$
END

Fig. B-4. CGT Controller Design

A21:	$\underline{A}_{21}$	(III-61d)
A12:	$\underline{A}_{12}$	(III-61b)
A22:	$\underline{A}_{22}$	(III-61e)
A13:	$\underline{A}_{13}$	(III-61c)
A23:	$\underline{A}_{23}$	(III-61f)
KXM:	$\underline{K}_{xm}$	(A-24a)
KXU:	$\underline{K}_{xu}$	(A-24b)
KXN:	$\underline{K}_{xn}$	(A-24c)

The controller gains ("KXM", "KXU", and "KXN") are also printed directly at the user terminal. Note that arrays $\underline{A}_{13}$, $\underline{A}_{23}$, and $\underline{K}_{xn}$ exist only if disturbance states are specifically modeled in the design model ($\underline{n}_d$ of Equations (B-1a,b)).

When execution of the CGT design computations is complete, the "controller evaluation" set of routines is automatically executed. These are discussed in the next subsection.

B.8.7 Controller Evaluation ("E")

A single set of routines performs the controller evaluation for both the PI and CGT designs. For the PI controller, the poles of the closed-loop discrete-time system matrix $\underline{\Phi}_{\delta CL}$ (Equation (A-18)) are computed and mapped into the primary strip in the continuous-time domain (the z-plane poles are not listed in output). These mapped closed-loop poles are printed to both the user terminal and the LIST file. The primary evaluation tool for both controllers is a time-response simulation. For the PI controller, the response is taken for non-zero initial conditions (IC's) on the states; for the CGT controller the response is given for a step input on any of the command model's inputs. In either case, the system dynamics can be propagated using the design model or truth model state transition equations. Time response runs may be run repeatedly for a specific controller design.

The I/O, logic, and computations involved in the controller evaluation are shown in Figure B-5. Decision blocks labeled "CGT" test for the type of controller being evaluated (CGT or PI). Decision blocks labeled "LTEVAL" test if the truth model is being used to propagate system dynamics (if not, the design model is being used).

The first prompt of the controller evaluation computational element asks if the evaluation is to be conducted with respect to truth model dynamics. If yes, the truth model may be established or modified (if previously established) in the manner described in Section B.8.3 above. Note that, since the continuous-time representation of the truth model is not preserved within CGTPIQ, "modifications" actually entails complete redefinition of the model. In the case that the truth model exists on the DATA file and only specific array elements are to be modified, it is convenient simply to read the truth model from the DATA file and modify matrix elements as shown in Figure B-2c. If the truth model had been established previously and no modification to it is desired, the existing discrete-time representation of the truth model is used. The design model is used to propagate

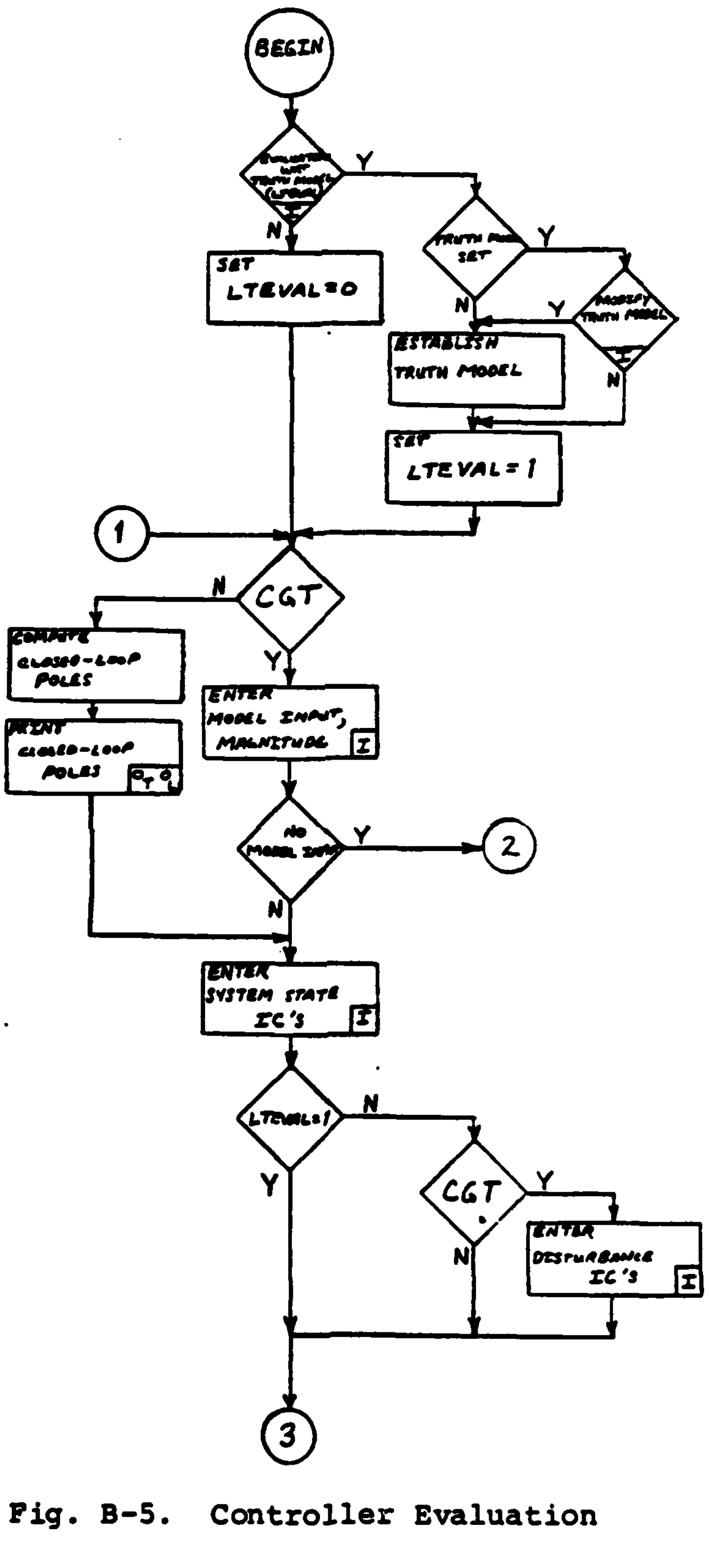


Fig. B-5. Controller Evaluation

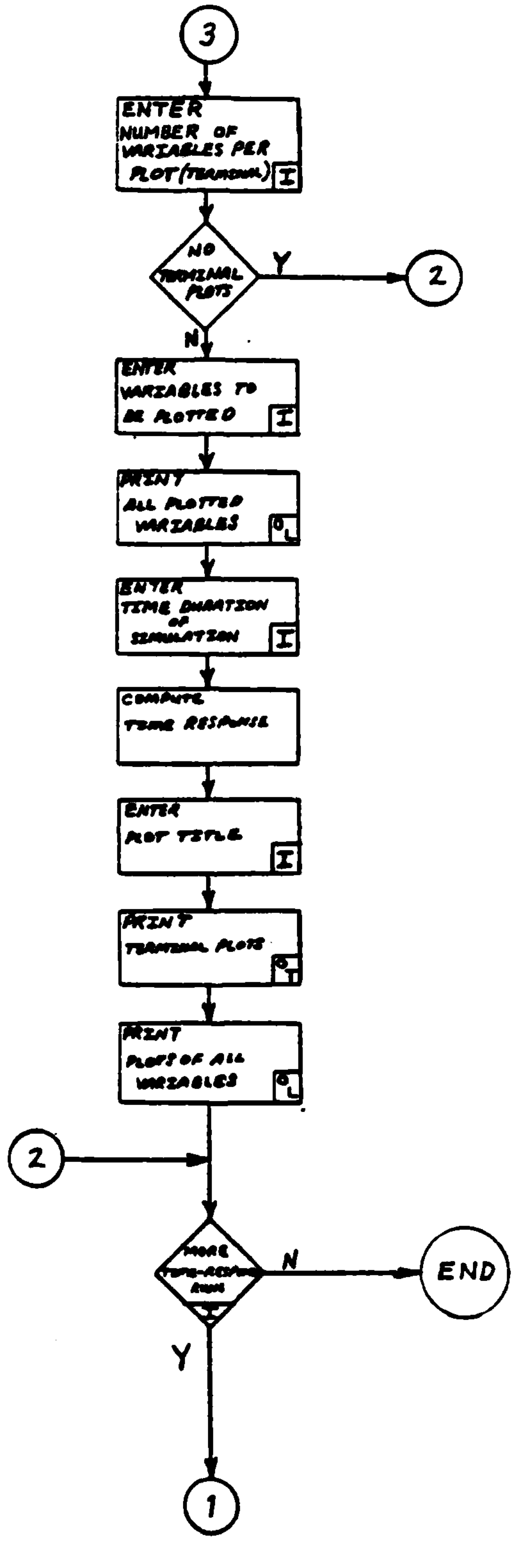

3
ENTER NUMBER OF VARIABLES PER PLOT (TERMINAL) I
NO TERMINAL PLOTS
Y
2
N
ENTER VARIABLES TO BE PLOTTED I
PRINT ALL PLOTTED VARIABLES O_L
ENTER TIME DURATION OF SIMULATION I
COMPUTE TIME RESPONSE
ENTER PLOT TITLE I
PRINT TERMINAL PLOTS O_T
PRINT PLOTS OF ALL VARIABLES O_L
2
MORE TIME-RESPONSE RUNS
N
END
Y
1

Fig. B-5--Continued

system dynamics if the truth model evaluation is not selected.

For the CGT evaluation, the first input prompt is for the index of the command input vector to which a step input is to be applied, and the magnitude of that step (only one command input is allowed at a time). CGTPIQ tests the input index for validity (within vector length bounds); if it is invalid the prompt is repeated. If the index is zero (or negative) the input is not accepted and the user is queried as to whether time-response runs are desired. If no time-responses are to be run, the controller evaluation routines are exited; otherwise, the prompt requesting command model input specification is repeated. If the CGT controller response is to be run, initial conditions on the system states may be entered. If the design model is used for evaluation and disturbance states exist in the model, they may be given initial conditions also.

For the PI evaluation the first input prompt is for initial conditions for the system states. The states that are actually given initial values are those of the design or truth model, according to the model used for propagation of dynamics. Initial conditions are entered for either controller in the same manner. Entry is of the index of the state within its appropriate state vector (design or truth model, or disturbance state vectors) and its initial value. Tests and termination of entry are as described in Section B.8.2 for multiple entries.

B.8.7.1 Time Response Plots

Time-response plots are of the "line printer" type and are output both to the terminal and to the LIST file. As many as two plots, each with as many as five dependent variables, may be printed at the user terminal. CGTPIQ prompts the user to specify the number of variables for each of the two plots (the user is to enter two integers). If the user enters non-positive integers for both plots, then a prompt queries the user as to whether time-response runs are desired. If no time-responses are to be run, the routines are exited. Note that when no plots are selected for terminal printing, none are output to LIST either and no time-responses are simulated.

In the case that plots are to be printed at the terminal, a series of prompts allow the user to specify exactly which variables shall be included. Variables are selected by specifying a name of the vector type for that variable and its index in two entries for each variable. The names of the vectors are:

X:	system state vector
Y:	system output vector
U:	system input vector
D:	disturbance state vector
M:	command model output vector

The system state vector is that of the design or truth models, according to the model used for propagation of dynamics. For example, the pair of entries "U" and "1" specifies that element 1 of the input vector U is to be plotted (note that "entry" includes a carriage return). The input prompt includes these definitions and includes only those variables relevant to the controller being evaluated. The model output and disturbance state vectors are only offered for CGT evaluations, and for the latter also only if the disturbance states are explicitly modeled and the design model propagates dynamics of the time-response (since for the truth model the system state vector includes all disturbance states which are considered to act on the system). Each user entry is tested for valid (and relevant) name and for valid index. Prompts specify the plot number and output number for each requested entry.

The user next is prompted to enter the time duration for the simulation. However, the duration actually simulated may be adjusted by the program: a time span that is the nearest integer multiple of 100 times the controller sample period is selected. Plots to the LIST file include the entire time span and use 100 equal time interval samples. Plots to the terminal include 50 time samples selected as follows: if the time duration specified by the user is less that 50 times the controller sample period, the samples plotted are the first 50 time samples from the simulation: otherwise, the entire time span is included in 50 equal-interval samples. Thus, for example, with a controller sample period of .02 seconds, a user-specified time duration of less than 1. second would yield plots to the terminal running from time=0. to time=1. and with .02 seconds between each sample; plots to the line printer from

LIST would include samples at the same interval but extending to time=2.

After completing the time-response simulation, a prompt requests a title for the plot and prescribes the field width available for the entry (50 characters). The title is applied to all plots generated from the single simulation.

Plots are printed with the independent (time) axis running vertically along the length of the output page with the origin at the top. Each sample time is identified along the left margin of the plot. The dependent axis is horizontal. Each variable is marked with an integer from 1 to 5 at each sample time. Note that since only one character can be printed in each location of the plot field, when two or more variables would occupy a single print position at a sample time, only the symbol of larges value (1 to 5) is printed. For plots to the terminal, if a model output is among the variables of a plot, then all variables in the plot are plotted over a single scale range to facilitate comparison of actual and desired output responses. In all other cases every variable plotted is scaled over its own range independently in order to achieve greater resolution for each in the plot field. The scale(s) are listed along the bottom of the plot. Rotation of the output page through 90 degrees in a counter-clockwise sense gives the usual abscissa-ordinate orientation. Terminal plots are 50 print positions wide; plots to LIST are 100 print positions wide.

Plots of all relevant variables in a time-response simulation (all states, inputs, and so on) are automatically output to the LIST file if terminal plots are requested. Five variables are included in each plot. A list identifying all the variables by type and index for each plot number and plot symbol is written to LIST prior to the plots.

When all plots have been printed, a prompt queries the user as to whether additional time-response runs are desired. If more are wished, the entire set of plotting options is repeated and the same controller may be evaluated under different conditions and/or different variables may be plotted. If no additional simulations are wished, the controller evaluation routines are exited.

B.8.8 Kalman Filter Design ("F")

The Kalman filter design routines compute the steady-state Kalman filter gains for the design model. Figure B-6 shows the I/O, logic, and computations involved. Note that the first execution of the filter design path bases its filter computations on the noise strengths specified upon initial entry of the design model. In subsequent executions, any of the noise strengths may be modified. The noise strengths are entered as vectors of the matrix diagonals (only diagonal elements may be modified). The matrices are

"STATE NOISE STRENGTHS": $\underline{Q}$ (B-2a)

"DISTURBANCE NOISE STRENGTHS": $\underline{Q}_n$ (B-2b)

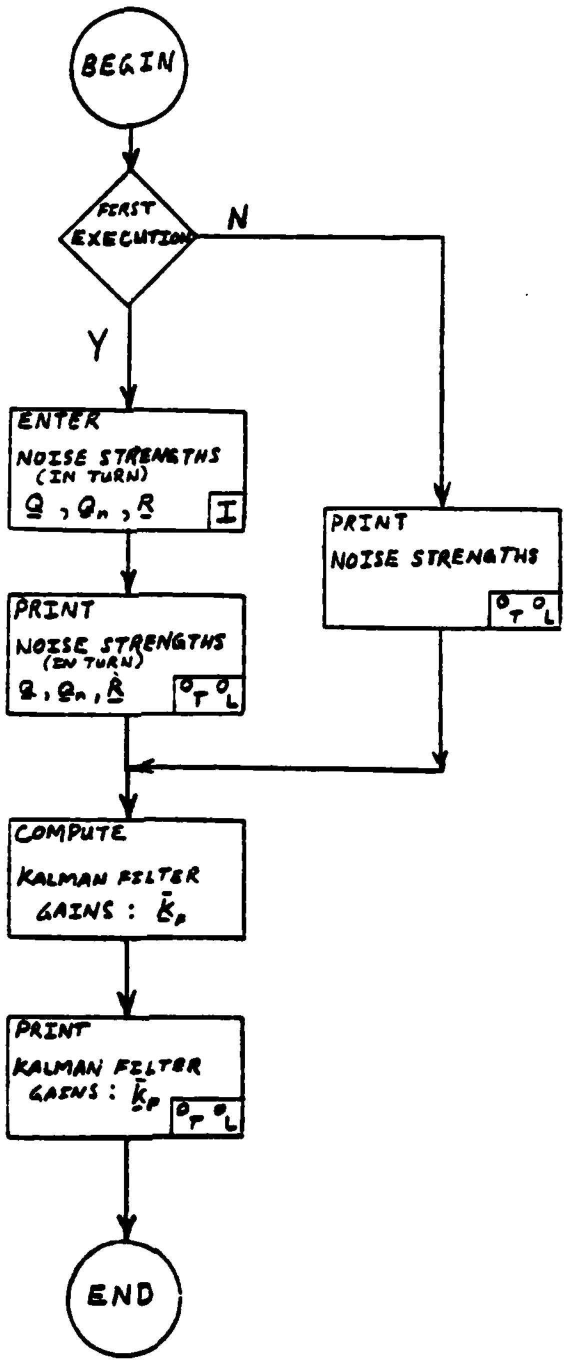

BEGIN
FIRST EXECUTION
N
Y
ENTER NOISE STRENGTHS (IN TURN) $\underline{Q}$, Q_n, $\underline{R}$
I
PRINT NOISE STRENGTHS
O_T O_L
PRINT NOISE STRENGTHS (IN TURN) $\underline{Q}$, $\underline{Q}_n$, $\underline{R}$
O_T O_L
COMPUTE KALMAN FILTER GAINS : $\bar{K}_p$
PRINT KALMAN FILTER GAINS : $\bar{K}_{cp}$
O_T O_L
END

Fig. B-6. Kalman Filter Design

"MEASUREMENT NOISE STRENGTHS":$\underline{R}$ (B-2c)

Prompts for state or disturbance noise strengths are given only if the design model specifies driving noises for the respective process dynamics. Negative noise strengths are not accepted.

In each execution of the filter design path the entire noise strength matrices and Kalman filter gain matrix are output to the terminal and LIST file. However, only the diagonal elements of the noise strength matrices are printed at the terminal.

Following computation of the filter gains, the Kalman filter design routines are exited. Execution proceeds automatically to the filter evaluation computational element described in the next subsection.

B.8.9 Filter Evaluation ("G")

Figure B-7 shows the I/O, logic, and computations of the filter evaluation routines. Execution of the filter evaluation requires that the system truth model be established, since the covariance analysis is performed with respect to the truth model. Comments in Section B.8.7 dealing with establishing the truth model apply equally in this set of routines, except that here use of the truth model is not optional.

Evaluation begins with computation of the eigenvalues of the system-filter matrix $\underline{\Phi}_{KF}$ (Equation (D-6)). As for the closed-loop PI regulated system, the discrete-time eigenvalues are mapped to the primary strip in

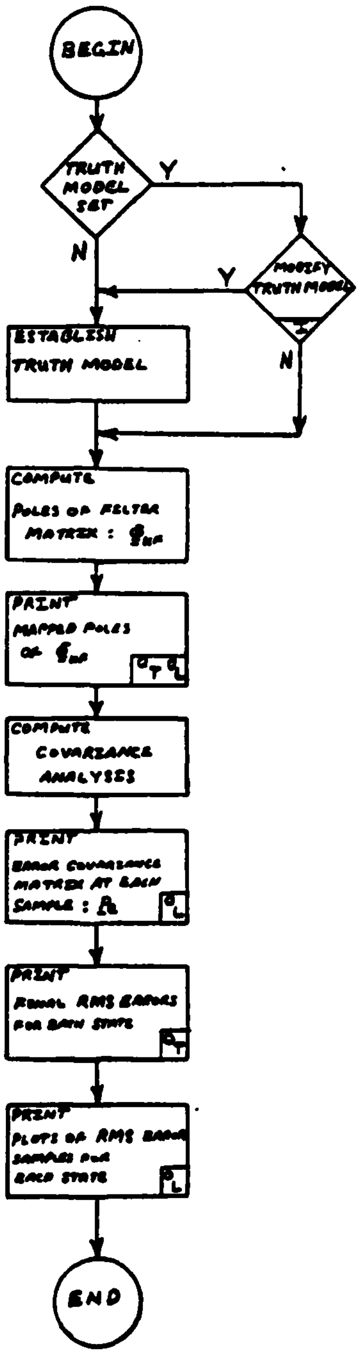

BEGIN
TRUTH MODEL SET
Y
N
MODIFY TRUTH MODEL
Y
N
ESTABLISH TRUTH MODEL
COMPUTE POLES OF FILTER MATRIX : Φ_{KF}
PRINT MAPPED POLES OF Φ_{KF}
O_T O_L
COMPUTE COVARIANCE ANALYSIS
PRINT ERROR COVARIANCE MATRIX AT EACH SAMPLE : P_k
O_L
PRINT FINAL RMS ERRORS FOR EACH STATE
O_T
PRINT PLOTS OF RMS ERROR SAMPLES FOR EACH STATE
O_L
END

Fig. B-7. Filter Evaluation

the continuous-time domain. These mapped poles are printed at the user terminal and output to the LIST file (the z-plane eigenvalues are not printed).

The primary evaluation tool applied to the filter design is a steady-state covariance analysis. The covariance matrix of the estimation errors of the filter using measurements of the truth model dynamics is propagated for 50 filter (controller) sample periods. Samples are coincident with the filter's sample times and the total number is fixed at 50. At each time sample the standard deviations of these "true" errors are computed as the square-roots of the diagonal elements of the error covariance matrix. The filter's own computed error covariance matrix, because of the steady-state assumption, is constant for all time samples. Taking the square-roots of the diagonal elements then gives the filter's estimate of the standard deviations of its errors in state estimation. Plots for each state are output to the LIST file showing the "true" and "computed" RMS errors for the 50 time samples. A title may be entered to be applied to all plots from the covariance analysis. In addition, the "true" and "computed" RMS errors at the final time sample are printed at the terminal.

This completes the filter evaluation. A new filter design may then be pursued, or any other design option may be selected.

B.8.10 Performance Evaluation Data ("H")

In performance evaluation data, the performance data determined throughout the program is written to the SAVE file (via the user's response to input prompts) for use in the performance evaluation accomplished via the program "PFEVAL" (Ref 21). (As a special note, if this data is to be used in the program "PFEVAL", the SAVE file must be transferred to a DATA file as discussed previously). Decision blocks "filter designed yet" and "CGT/PI designed yet" correspond to program decision "flags". Note that performance data may be written to the SAVE file from either the controller or the filter design path.

B.9 Program Messages

A variety of messages may be printed at the terminal and/or output to the LIST file during program execution. Some are purely informational, are clear in their meaning, and provide no essential insight into progress of the design or possible difficulties in program execution. Such messages are not discussed here. The remaining messages relate to errors or potential design difficulties and are considered in categories of memory allocation, dimensional errors, or computational problems.

B.9.1 Memory Allocation

CGTPIQ uses vectors in named Commons for array storage. These vectors are dimensioned in the main routine and a variable in the Common is set to the value allocated.

These vectors are then partitioned within CGTPIQ to store individual arrays. Before storing arrays into each vector in Common, the storage needed is computed. In the case of temporary storage vectors, at each point in execution at which a new allocation is needed, the required storage is recomputed. If more memory will be needed in a vector than has been allocated, a message is written specifying the name of the Common and the necessary minimum allocation. A typical message of this type is

"INSUFFICIENT MEMORY /SYSMTX/, NEED: nnnn"

in which the common /SYSMTX/ has too little storage for the problem and 'nnnn' is the dimension required for the vector in that common. For the vectors containing the dynamics models, the model with insufficient memory is identified by name. The Commons for the design, truth, and command models are /DSNMTX/, /TRUMTX/, and /CMDMTX/, respectively. After printing such a message to the user terminal, execution is aborted.

In its existing form, CGTPIQ will have sufficient vector allocations to deal successfully with problems of many different combinations of dimensions and with system matrices in the range of 10 to 20 states. Since the program will not allow allocated memory to be exceeded, it is reasonable to attempt any given problem and let the program either deal with it successfully, or let it write the appropriate memory message if the problem cannot be accommodated. (Section A.9 of the programmer's manual of the thesis by Floyd (Ref 9) discusses the steps necessary to obtain a new CGTPIQ with different memory allocations)

B.9.2 Dimensional Errors

As each of the dynamics models is established, or just prior to computations which assume relations among the design and command system and disturbance state dimensions (CGT computations), the dimensional constraints mentioned previously are tested. In terms of the dimension notation of Equations (B-4), (B-8), and (B-11) the constraints are

Design model:	$p=r$ and $n\geq d$
Truth model:	$r_T=r$ and $m_T=m$
Command model:	$p_M=p$ and $n\geq n_M$
	$p=n_M$ (for implicit design only)

If such a constraint is not satisfied, a message is written to the user terminal identifying the problem and execution is aborted. When the constraint affects only a specific section of code, or if redefinition of the model (command or truth) can resolve the error, then only the affected execution path is aborted. In other cases, execution is aborted completely.

Other dimensional tests are made in the Kalman filter design and evaluation computational elements. For filter design, it is necessary that the system state and disturbance state driving noise dimensions not both be equal to zero, and that the number of measurements be non-zero. These are constraints on the design model:

$$\begin{bmatrix} w>0 \\ \text{or} \\ w_D>0 \end{bmatrix}$$

and

$m>0$

For filter evaluation, the number of driving noises for the truth model must be non-zero:

$w_T>0$

since in the case of $w_T=0$, the system is deterministic and the covariance analysis would not provide much useful information for evaluation of the filter's performance. If any of these constraints are not satisfied, a message is written to the user terminal identifying the problem and execution of the filter design-evaluation computational elements is aborted.

B.9.3 Computational Problems

In certain of the computations, characteristics of the particular design problem may be identified as having potential impact on the attainment of design objectives. Messages identifying these characteristics may be considered informational. Other messages describe computational problems that are immediately fatal.

In computing the Pi matrix of Equation (II-18), a pair of messages may be generated to the LIST file:

"PI MATRIX IS RANK DEFECTIVE"

and

"nr X nc MT RANK mr"

in which "nr" and "nc" are the row and column dimensions of the Pi matrix and "mr" is it rank. The first message is also printed at the user terminal. The equations employing Pi assume it to be an ordinary matrix inverse. If it is rank defective, the matrix pseudo-inverse is computed instead. Execution of the program continues since useful results may still be obtained through use of the pseudo-inverse (Ref 18:124).

Solution of the Riccati equations for the PI controller and the Kalman filter is achieved using an iterative algorithm (Ref 12) which may generate messages of information or fatal error. The information message for the PI is,

"RICCATI SOLN IS PSD--RANK mr"

in which "PSD" means positive semi-definite. For the Kalman filter the corresponding message is,

"OBSERVABILITY MATRIX IS nr X nc OF RANK mr"

in which "nr", "nc", and "mr" are the row, column dimensions and the rank, respectively. These messages convey the same information concerning system observability. The message is written in the case of the PI Riccati equation only if the solution is positive semi-definite (rank defective). Both messages are output to the LIST file only.

For both the PI and filter Riccati equations fatal error messages are identical:

"RICCATI NON-CONVERGENT IN nn ITERATIONS"

or

"RICCATI BLOW UP AT ITERATION nn INITIAL N = mm" in which "nn" is the iteration counter at the occurrence of the error and "mm" is the value of a variable set internally and used in achieving initialization of the iterative sequence (the internal variable is not available for modification by the user). The first message indicates that the sequence of iterates did not converge. The second message may indicate numerical difficulties or uncontrollability (unobservability) of the system of the PI (filter) equations. Both messages are output to the LIST file only; a system error exit routine is then called which writes "EXIT" to the user terminal and aborts program execution. Note that in the event of such an abort, the local files SAVE, DATA, and LIST are not rewound automatically.

In computing the CGT controller gains an error may occur in solving for the matrix partitions $\underline{A}_{11}$ or $\underline{A}_{13}$ (Equation (III-61)). If the iterative refinements to these solutions do not converge to within the established tolerance (1.E-6), then the following message is written both to the terminal and the LIST file:

"SOLUTION ERROR FOR 'A' (CGT) AFTER 3 ITERATIONS = nnn"

in which "nnn" is the Euclidean norm of the refining matrix solution (residual) at the last iteration. The message is considered to be infomational, and execution proceeds normally. However, in case the value of the residual norm is large compared to the convergence tolerance, the CGT design solution can be expected to be invalid.

B.10 CGTPIQ Segmentation Job Control

The following listing shows the job control commands and segmentation directives used in obtaining a segmented object file suitable for interactive execution on the CDC CYBER computer system. The job employs three object files: "L", "S", and "A". The routines on each of these files are (see the program listing)

"L": 'MAIN' and all optional routines('MAIN' through 'TBLUP1')

"S": 'CGTPIQ SUBS' ('CGTXQ' through 'VARSCL')

"A": 'LIBRARY'

Object files L and S are loaded into memory in the order of MAIN, then CGTPIQ SUBS. The "NOGO" command then completes the load from LIBRARY and system routines in order, but does not initiate execution. Next, the segmentation directives are executed (segmentation directives appear between the pair of "*EOR" lines). When segmentation is complete, the resulting object file is cataloged.

In this listing, the names given the various object files ("L", "S", "A") are arbitrary, and the "ATTACH" commands may occur in any order. The file name ("lfn") given in the "LIBRARY,lfn" command must correspond to the name used in attaching the object file of 'LIBRARY' routines. The name given to the segmented object file is arbitrary but must be consistent in the "REQUEST", "SEGLOAD", and "CATALOG" commands. The segmentation directives should not be modified in any way.

```
100=JPM. T830287,MCMILLIAN
110=MAP,FULL.
120=ATTACH,CGTX.
130=ATTACH,DLKLIB,ID=T820303.
140=LIBRARY,DLKLIB.
150=REQUEST,CGTPIQ,*PF.
160=SEGLOAD(B=CGTPIQ)
170=LOAD(CGTX)
180=NOGO.
190=REWIND,CGTPIQ.
200=CATALOG,CGTPIQ,CGTPIQ.
210=*EOR
220=SETUP        INCLUDE DSCRT
230=SREGPI       INCLUDE RQWGTS,MLINEQ,SYMFCT
240=FLTRK        INCLUDE RQWGTS,KFLTR,MLINEQ,SYMFCT,INTEG
250=STRTH        INCLUDE DSCRTT,INTEG
260=SDSN         INCLUDE QDSCRT
270=CEVAL        INCLUDE PLOTLP,VARSCL,RPLOTF,WPLOTF,STRPLT
280=FEVAL        INCLUDE PLOTLP,VARSCL,RPLOTF,WPLOTF,STRPLT,DACOV
290=B1           TREE SETUP-(SDSN,SCMD,STRTH)
300=B2           TREE PIMTX
310=B3           TREE SREGPI
320=B4           TREE SCGT
330=B5           TREE CEVAL
340=B6           TREE FLTRK
350=B7           TREE FEVAL
360=B8           TREE PFDATA
370=A            TREE CGTXQ-(B1,B2,B3,B4,B5,B6,B7,B8)
380=ROOT         TREE MAIN-A
390=             GLOBAL MAIN1,MAIN2,INOU,DESIGN,FILES,SYSMTX,ZMTX1,ZMTX2,
400=,NDIMD,LOCD,DSNMTX,NDIMC,LOCC,CMDMTX,NDIMT,LOCT,TRUMTX,LCNTRL,CONTROL,
410=,LREGPI,CREGPI,LCGT,CCGT,LKF,CKF,AMC,BDG
420=           END
430=*EOR
440=*EOF
```

B.11 Running CGTPIQ

This "User's Guide" assumes that a segmented executable object file of CGTPIQ exists according to the previous section. For an existing CGTPIQ object file in the permanent file system, the following commands must be entered in INTERCOM to run the program:

```
CONNECT,INPUT,OUTPUT
ATTACH,CGTPIQ,pfn
CGTPIQ
```

in which 'pfn' is the permanent file on which the object file is cataloged. CGTPIQ will then execute as described in Section B.8. Additional commands before and after execution may be appropriate according to one's intended use of the various local files which CGTPIQ employs during execution (like attaching a DATA file).

B.12 CGTPIQ Input/Output Listing

The input/output (I/O) listing given at the end of this section is from a single execution of CGTPIQ. It shows a CGT design, a PI controller design, a CGT/PI controller design, and a Kalman filter design for a simple design model. Details concerning the design, truth, and command models employed are presented in Section B.12.2. The models were selected to test the program thoroughly and give the user a good feel for program execution. The models are otherwise meaningless in the real world context. The I/O shown is that obtained directly at the user terminal during execution. The listing is complete and in order. The additional information written to the LIST file is not presented here but the last section of this guide gives a brief description of the output appearing on a LIST file.

Prior to and following program execution, INTERCOM prompts for input are given by "COMMAND-". Within program execution entries occur in two ways: (1) when the entry is on the same line as an input prompt, the entry is bounded on the left by the symbol ">"; (2) in case of multiple entries for a single prompt, the second and subsequent entries are not prompted but are entered on the next lines, one entry per line.

Portions of I/O are discussed within individual numbered paragraphs. Each portion of listing is identified by a number in parentheses at the top center of the page where it begins, and these numbers correspond to the paragraph numbers below.

B.12.1 <u>Summary of Input/Output</u>

(1) Following "LOGIN", the executable object file 'CGTPIQ' is attached. Next, permanent file space is requested for the local file name 'SAVE'; the SAVE file will be generated during subsequent execution of the program. The "CONNECT" command defines the user terminal as the device that communicates through the FORTRAN standard 'INPUT' and 'OUTPUT' files. Program execution is initiated with the simple command "CGTPIQ", which loads the local file CGTPIQ and begins execution at its starting address.

(2) Program execution begins with output of an

identifying header which includes the current date and time (obtained from a call to system real-time clock routines). The first user entry is the sample period of the controller. Next, the design model is established under prompt from the terminal. The design model is then written to the SAVE file. The poles (eigenvalues) of the design model ($\underline{A}$ matrix) are automatically computed and printed.

(3) The controller design path is then pursued with selection of an open-loop CGT design requested (since there are no PI gains yet). The command model is then input, written to the SAVE file, and the poles of the command matrix ($\underline{A}_m$) automatically output. An informational error message is then printed. This does not affect operation but the validity of the solution is in question (see Section B.9).

(4) The open-loop CGT gains (KXM, KXU and KXN) are then computed and printed. The evaluation is chosen with respect to the truth model, so the truth model is input next and written to the SAVE file. The poles of the truth matrix ($\underline{A}_t$) are automatically computed and printed.

(5) A step of magnitude one is input on model input one. There are no initial conditions (IC) placed on the states. One plot of two variables is selected with Y1 and M1 being the variables to be plotted. A plot duration of one second is chosen and a title given to the plot. The open-loop CGT time response is then plotted at the terminal.

(6) A PI design is chosen next and implicit model following requested, but, since the command model state

dimension did not equal the design model output dimension, this request was aborted (another command model would need to be input to use the implicit path). The explicit weights were then input and displayed with the (2,2) element of the $\underline{X}$ matrix being modified. (Note that the $\underline{X}$ matrix is computed anew each iteration from the weighting matrices on the outputs and on the integral of the regulation error. Thus, modifications made explicitly to $\underline{X}$ are not preserved between design iterations.)

(7) The PI gains GC1 and GC2 are computed and displayed followed by the printout of the continuous-time mapped poles of the closed loop system matrix (note the lack of one complex complementary pole of the closed-loop system matrix: when this happens it usually indicates a poor choice of quadratic weights, sampling time, etc., and a potentially useless design, and is caused by a pole in the right-half z-plane not being able to be mapped to the s-plane). Initial conditions of one are placed on states one and two, and one plot of three variables is requested with X2, Y1 and U1 being the variables to be plotted. A plot duration of one second is requested and a title given to the plot. The time response plot of the PI controller is then presented.

(8) A closed-loop CGT/PI is then selected since the PI gains have been computed. The CGT/PI gains (KXM, KXU and KXN) are computed and printed. Controller evaluation with respect to the truth model is selected, a step input of magnitude one placed on model input one, and zero initial

conditions placed on the states. One plot of two variables is requested with Y1 and M1 chosen as the variables. A duration of one second is chosen and a plot title is given. The closed-loop CGT/PI time response is then plotted at the terminal. Note the difference in performance between the open-loop CGT and the closed-loop CGT/PI controllers.

(9) No additional time-responses of the CGT/PI controller are requested. The Kalman filter path is then selected. Since this is the first execution of the Kalman filter design, the noise strength matrices ($\underline{Q}$ and $\underline{R}$) specified in the existing definition of the design model are used to compute the Kalman filter gain matrix (note that in subsequent iterations of the Kalman filter design path, the user may modify the diagonal elements (only) of the noise strength matrices). The diagonal elements of the noise strength matrices and the entire filter gain matrix are printed. Next, a title is entered to apply to the plots of state estimation error standard deviations (output to LIST file only). The final values of true and filter computed RMS errors for the design model state estimates are printed. The performance evaluation data is then written to the SAVE file. (This is the information needed by the program "PFEVAL".)

(10) The filter design is not repeated, nor are any of the other designs. Upon terminating execution, the PI gains determined previously are written to the SAVE file. The command "FILES" gives a listing of existing files. Note that files SAVE, LIST and PLOT have been generated

automatically during program execution. The SAVE file (containing the design, truth, and command model data as well as the PI gains GC1 and GC2, and the Kalman filter performance data) is then cataloged for future use (as a DATA file). The just-created SAVE file is then copied to the local file named DATA. Next, the LIST file is sent to a line printer for listing. Finally, the SAVE file is returned, making re-execution of the program feasible (since SAVE had been made a permanent file, it could not be written to in subsequent executions). In a subsequent execution, the various dynamics models and the PI gains would be available from the new DATA file and a new SAVE file would be created (if desired). (Note: The required parameters to run the evaluation program "PFEVAL" are also written to the SAVE file.) However, in this case there is no repeated execution of the program, and the user enters a "LOGOUT" from the system.

B.12.2 Simple Design Problem

<u>Design Model</u>

$$\dot{\underline{x}}(t) = \begin{bmatrix} -2 & 5 \\ -5 & 2 \end{bmatrix} \underline{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \underline{u}(t) + \begin{bmatrix} 3 & -1 \\ 1 & 5 \end{bmatrix} \underline{n}_d(t) + \begin{bmatrix} .1 \\ .1 \end{bmatrix} \underline{w}(t)$$

$$\underline{Q} = 1$$

$$\dot{\underline{n}}_d(t) = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \underline{n}_d(t) + \begin{bmatrix} .1 \\ .1 \end{bmatrix} \underline{w}_d(t)$$

$$\underline{Q}_n = 1$$

$$\underline{y}(t) = [1 \quad 0]\underline{x}(t) + [1]\underline{u}(t) + [1 \quad 1]\underline{n}_d(t)$$

$\underline{z}(ti) = [1 \quad 0]\underline{x}(ti) + [0 \quad 1]\underline{n}_d(ti) + \underline{v}(ti)$

$\underline{R} = .1$

Truth Model

$$\dot{\underline{x}}_t(t) = \begin{bmatrix} -2 & 5 & 3 & 1 \\ -5 & -2 & 1 & 5 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \underline{x}_t(t) + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \underline{u}_t(t) + \begin{bmatrix} .1 \\ .1 \\ 0 \\ 0 \end{bmatrix} \underline{w}_t(t)$$

$\underline{Q}_t = 1$

$\underline{z}_t(t_i) = [1 \quad 0 \quad 0 \quad 1]\ \underline{x}_t(t_i) + \underline{v}_t(t_i)$

$\underline{R}_t = .1$

$$\underline{x}'(t) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \underline{x}_t(t)$$

$$\underline{n}_d'(t) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \underline{x}_t(t)$$

Command Model

$$\dot{\underline{x}}_m(t) = \begin{bmatrix} -5 & 5 \\ -5 & -5 \end{bmatrix} \underline{x}_m(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \underline{u}_m(t)$$

$\underline{y}_m(t) = [10 \quad 0]\underline{x}_m(t) + [1]\underline{u}_m(t)$

```
DATE  11/09/83
TIME  21.25.21.

PLEASE LOGIN
LOGIN,T830287

XXXXXXXXXX ENTER PASSWORD-


11/09/83   LOGGED IN AT  21.26.08.
           WITH USER-ID X3
           EQUIP/PORT 14/021
LOGIN       CREATED 11/09/83  TODAY IS 11/09/83

COMMAND- BEGIN,CLASS,,MACM,JPM

COMMAND- CONNECT,INPUT,OUTPUT

COMMAND- GET,CGTPIQ,ID=MACM

 FILE NAME CGTPIQ     HAS BEEN RETRIEVED
COMMAND- REQUEST,SAVE,*PF

COMMAND- CGTPIQ
```

```
                    * * * CGTPIQ * * *
        PROGRAM TO DESIGN A COMMAND GENERATOR TRACKER
   USING A REGULATOR WITH PROPORTIONAL PLUS INTEGRAL CONTROL
        BASED ON THE INTEGRAL OF THE REGULATION ERROR,
          AND A KALMAN FILTER FOR STATE ESTIMATION.
                    * * * CGTPIQ * * *

       DATE :  11/09/83

       TIME :  21.28.36.



ENTER SAMPLE PERIOD FOR DIGITAL CONTROLLER >.02

READ DESIGN  MODEL FROM 'DATA' FILE (Y OR N) >N

ENTER DESIGN  MODEL FROM TERMINAL (Y OR N) >Y

ENTER N  >2

ENTER R  >1

ENTER P  >1

ENTER M  >1

ENTER D  >2

ENTER W  >1

ENTER WD >1
ENTER A
ENTER I,J AND M(I,J)--(0/ WHEN COMPLETE) :  2 BY  2 >1 1 -2
1 2 5
2 1 -5
2 2 -2
0/


ENTER B
ENTER I,J AND M(I,J)--(0/ WHEN COMPLETE) :  2 BY  1 >2 1 1
0/


ENTER EX
ENTER I,J AND M(I,J)--(0/ WHEN COMPLETE) :  2 BY  2 >1 1 3
1 2 -1
2 1 1
2 2 5
0/


ENTER G
ENTER I,J AND M(I,J)--(0/ WHEN COMPLETE) :  2 BY  1 >1 1 .1
2 1 .1
0/


ENTER Q
ENTER I,J AND M(I,J)--(0/ WHEN COMPLETE) :  1 BY  1 >1 1 1
0/


ENTER C
ENTER I,J AND M(I,J)--(0/ WHEN COMPLETE) :  1 BY  2 >1 1 1
0/


ENTER DY
ENTER I,J AND M(I,J)--(0/ WHEN COMPLETE) :  1 BY  1 >1 1 1
0/


ENTER EY
ENTER I,J AND M(I,J)--(0/ WHEN COMPLETE) :  1 BY  2 >1 1 1
1 2 1
0/


ENTER H
ENTER I,J AND M(I,J)--(0/ WHEN COMPLETE) :  1 BY  2 >1 1 1
0/
```

```
ENTER HN
ENTER I,J AND M(I,J)--(0/ WHEN COMPLETE) :  1 BY  2 >1 2 1
0/


ENTER R
ENTER I,J AND M(I,J)--(0/ WHEN COMPLETE) :  1 BY  1 >1 1 .1
0/


ENTER AN
ENTER I,J AND M(I,J)--(0/ WHEN COMPLETE) :  2 BY  2 >1 1 1
1 2 1
2 1 1
2 2 1
0/


ENTER GN
ENTER I,J AND M(I,J)--(0/ WHEN COMPLETE) :  2 BY  1 >1 1 .1
2 1 .1
0/


ENTER QN
ENTER I,J AND M(I,J)--(0/ WHEN COMPLETE) :  1 BY  1 >1 1 1
0/


MODIFY MATRIX ELEMENTS (Y OR N) >N

WRITE DESIGN  MODEL TO 'SAVE' FILE (Y OR N) >Y

     DESIGN  MODEL WRITTEN TO 'SAVE' FILE

POLES OF DESIGN  MATRIX

      -2.0000000E+00  +J(  5.0000000E+00)
      -2.0000000E+00  +J( -5.0000000E+00)
```

(3)

```
CONTROLLER DESIGN (Y OR N) >Y

DESIGN REG/PI (Y OR N) >N

DESIGN CGT (Y OR N) >Y

READ REG/PI GAINS FROM 'DATA' FILE (Y OR N) >N

READ COMMAND MODEL FROM 'DATA' FILE (Y OR N) >N

ENTER COMMAND MODEL FROM TERMINAL (Y OR N) >Y

ENTER NM >2

ENTER RM >1

ENTER PM >1


ENTER AM
ENTER I,J AND M(I,J)--(0/ WHEN COMPLETE) :  2 BY  2 >1 1 -5
1 2 5
2 1 -5
2 2 -5
0/


ENTER BM
ENTER I,J AND M(I,J)--(0/ WHEN COMPLETE) :  2 BY  1 >2 1 1
0/


ENTER CM
ENTER I,J AND M(I,J)--(0/ WHEN COMPLETE) :  1 BY  2 >1 1 10
0/


ENTER DM
ENTER I,J AND M(I,J)--(0/ WHEN COMPLETE) :  1 BY  1 >1 1 1
0/

MODIFY MATRIX ELEMENTS (Y OR N) >N

WRITE COMMAND MODEL TO 'SAVE' FILE (Y OR N) >Y

     COMMAND MODEL WRITTEN TO 'SAVE' FILE

POLES OF COMMAND MATRIX

      -5.0000000E+00  +J(  5.0000000E+00)
      -5.0000000E+00  +J( -5.0000000E+00)

SOLUTION ERROR FOR 'A'(CGT) AFTER 3 ITERATIONS =   2.1370445E+01
```

```
KXM MATRIX

    33.44       -13.57

KXU MATRIX

   -.2812

KXN MATRIX

   -1.131       -1.484

CONTROLLER EVALUATION WRT TRUTH MODEL (Y OR N) >Y

READ TRUTH   MODEL FROM 'DATA' FILE (Y OR N) >N

ENTER TRUTH   MODEL FROM TERMINAL (Y OR N) >Y

ENTER NT >4

ENTER RT >1

ENTER MT >1

ENTER WT >1


ENTER AT
ENTER I,J AND M(I,J)--(0/ WHEN COMPLETE) :  4 BY  4 >1 1 -2
1 2 5
1 3 3
1 4 -1
2 1 -5
2 2 -2
2 3 1
2 4 5
0/


ENTER BT
ENTER I,J AND M(I,J)--(0/ WHEN COMPLETE) :  4 BY  1 >2 1 1
0/


ENTER GT
ENTER I,J AND M(I,J)--(0/ WHEN COMPLETE) :  4 BY  1 >1 1 .1
2 1 .1
0/


ENTER QT
ENTER I,J AND M(I,J)--(0/ WHEN COMPLETE) :  1 BY  1 >1 1 1
0/
```

```
ENTER HT
ENTER I,J AND M(I,J)--(0/ WHEN COMPLETE) :  1 BY  4 >1 1 1
1 4 1
0/


ENTER RT
ENTER I,J AND M(I,J)--(0/ WHEN COMPLETE) :  1 BY  1 >1 1 .1
0/


ENTER TDT
ENTER I,J AND M(I,J)--(0/ WHEN COMPLETE) :  2 BY  4 >1 1 1
2 2 1
0/


ENTER TNT
ENTER I,J AND M(I,J)--(0/ WHEN COMPLETE) :  2 BY  4 >1 3 1
2 4 1
0/


MODIFY MATRIX ELEMENTS (Y OR N) >N


WRITE TRUTH   MODEL TO 'SAVE' FILE (Y OR N) >Y

     TRUTH   MODEL WRITTEN TO 'SAVE' FILE

POLES OF TRUTH   MATRIX

      -2.0000000E+00  +J(  5.0000000E+00)
      -2.0000000E+00  +J( -5.0000000E+00)
       0.             +J(  0.           )
       0.             +J(  0.           )
```

```
ENTER MODEL INPUT AND STEP VALUE : 1 >1 1

ENTER STATE AND IC VALUE (0/ TERMINATES):  4 >0/

2 PLOTS OF 5 VARIABLES MAY BE PRINTED AT THE TERMINAL -- SPECIFY NUMBER
FOR EACH (N1,N2) >2 0

ENTER OUTPUTS BY TYPE AND INDEX IN 2 ENTRIES--TYPES ARE
STATE : 'X'
OUTPUT : 'Y'
INPUT : 'U'
MODEL : 'M'

PLOT  1
OUTPUT  1 >Y

          >1

OUTPUT  2 >M

          >1

ENTER TIME DURATION FOR RESPONSE, IN SECONDS >1

+---------- ENTER TITLE IN GIVEN FIELD ----------+
SIMPLE DESIGN TEST OF CGT
```

SIMPLE DESIGN TEST OF CGT

```
    0.00      +         +  1      +         2         +         +
     .02      +       1 +         +         2         +         +
     .04      +   1     +         +         2         +         +
     .06      + 1       +         +         +2        +         +
     .08      +1        +         +         + 2       +         +
     .10      1         +         +         + 2       +         +
     .12      +1        +         +         +  2      +         +
     .14      + 1       +         +         +    2    +         +
     .16      +  1      +         +         +     2   +         +
     .18      +: : 1 : :+: : : : :+: : : : :+: : : 2 :+: : : : :+
     .20      +      1  +         +         +       2 +         +
     .22      +         1         +         +        2+         +
     .24      +         + 1       +         +         2         +
     .26      +         +    1    +         +         +2        +
     .28      +         +      1  +         +         +2        +
     .30      +         +         1         +         + 2       +
     .32      +         +         + 1       +         +  2      +
     .34      +         +         +    1    +         +   2     +
     .36      +         +         +      1  +         +   2     +
     .38      +: : : : :+: : : : :+: : : : 1+: : : : :+: : 2 : :+
     .40      +         +         +         +1        +    2    +
     .42      +         +         +         +  1      +    2    +
     .44      +         +         +         +    1    +     2   +
     .46      +         +         +         +     1   +     2   +
     .48      +         +         +         +       1 +     2   +
     .50      +         +         +         +        1+     2   +
     .52      +         +         +         +         +1     2  +
     .54      +         +         +         +         + 1    2  +
     .56      +         +         +         +         +  1   2  +
     .58      +: : : : :+: : : : :+: : : : :+: : : : :+: :1: 2 :+
     .60      +         +         +         +         +   1  2  +
     .62      +         +         +         +         +    1 2  +
     .64      +         +         +         +         +     12  +
     .66      +         +         +         +         +     12  +
     .68      +         +         +         +         +      2  +
     .70      +         +         +         +         +      2  +
     .72      +         +         +         +         +      2  +
     .74      +         +         +         +         +      21 +
     .76      +         +         +         +         +      21 +
     .78      +: : : : :+: : : : :+: : : : :+: : : : :+: : : 21:+
     .80      +         +         +         +         +      21 +
     .82      +         +         +         +         +      2 1+
     .84      +         +         +         +         +      2 1+
     .86      +         +         +         +         +     2  1+
     .88      +         +         +         +         +     2  1+
     .90      +         +         +         +         +     2  1+
     .92      +         +         +         +         +     2  1+
     .94      +         +         +         +         +     2  1+
     .96      +         +         +         +         +     2  1+
     .98      +: : : : :+: : : : :+: : : : :+: : : : :+: : :2: 1+
    1.00      +         +         +         +         +     2 1 +
SCALE       -.8000    -.2000      .4000    1.0000    1.6000    2.2000
```

(6)

```
MORE TIME RESPONSE RUNS (Y OR N) >N

CONTROLLER DESIGN (Y OR N) >Y

DESIGN REG/PI (Y OR N) >Y

INCORPORATE IMPLICIT MODEL (Y OR N) >Y

MODIFY COMMAND MODEL (Y OR N) >N

COMMAND MODEL STATE DIM MUST EQUAL SYSTEM OUTPUT DIM

CONTROLLER DESIGN (Y OR N) >Y

DESIGN REG/PI (Y OR N) >Y

INCORPORATE IMPLICIT MODEL (Y OR N) >N

ENTER WEIGHTS ON OUTPUT DEVIATIONS:  1
ENTER I AND QW(I,I)--(0/ WHEN COMPLETE) >1 5
0/

ENTER WEIGHTS ON INTEGRAL OF REGULATION ERROR:  1
ENTER I AND QW(I,I)--(0/ WHEN COMPLETE) >1 20
0/

ENTER WEIGHTS ON CONTROL MAGNITUDES:  1
ENTER I AND QW(I,I)--(0/ WHEN COMPLETE) >1 1
0/


Y   MATRIX

    5.000

UQ  MATRIX

    20.00

MODIFY ELEMENTS OF 'X' MATRIX (Y OR N) >Y

LIST 'X' MATRIX TO TERMINAL (Y OR N) >Y

X   MATRIX

    5.000       0.           0.
   0.           0.           0.
   0.           0.            20.00
ENTER I,J AND M(I,J)--(0/ WHEN COMPLETE) :  3 BY  3 >2 2 10
0/
```

```
UM  MATRIX

    1.000

GC1 MATRIX

    1.006        4.9054E-02

GC2 MATRIX

    1.381

CONTROLLER EVALUATION WRT TRUTH MODEL (Y OR N) >Y

MODIFY TRUTH MODEL (Y OR N) >N

POLES OF REGPI   MATRIX

      -1.9740188E+00  +J(  5.4833537E+00)
      -1.9740188E+00  +J( -5.4833537E+00)
      -4.8082993E+01  +J(  1.5707963E+02)

ENTER STATE AND IC VALUE (0/ TERMINATES):  4 >1 1
2 1
0/

2 PLOTS OF 5 VARIABLES MAY BE PRINTED AT THE TERMINAL -- SPECIFY NUMBER
FOR EACH (N1,N2) >3 0

ENTER OUTPUTS BY TYPE AND INDEX IN 2 ENTRIES--TYPES ARE
STATE : 'X'
OUTPUT : 'Y'
INPUT : 'U'

PLOT  1
OUTPUT  1 >X

          >2

OUTPUT  2 >Y

          >1

OUTPUT  3 >U

          >1

ENTER TIME DURATION FOR RESPONSE, IN SECONDS >1

+---------- ENTER TITLE IN GIVEN FIELD ----------+
SIMPLE DESIGN TEST, PI
```

```
       SIMPLE DESIGN TEST, PI
   0.00      +         +         +         +     3   + 1       +
    .02      +3 2      +         +         +       1 +         +
    .04      + 3  2    +         +         +   1     +         +
    .06      3  2      +         +         1         +         +
    .08      3  2      +         +     1   +         +         +
    .10      3  2      +         +  1      +         +         +
    .12      3 2       +        1+         +         +         +
    .14      +32       +     1   +         +         +         +
    .16      +23       +  1      +         +         +         +
    .18      +2 :3: : :+1 : : : :+: : : : :+: : : : :+: : : : :+
    .20      +2    3 1 +         +         +         +         +
    .22      +2    1 3 +         +         +         +         +
    .24      2   1     3         +         +         +         +
    .26      2  1      +  3      +         +         +         +
    .28      2 1       +    3    +         +         +         +
    .30      21        +       3 +         +         +         +
    .32      2         +         +3        +         +         +
    .34      2         +         +   3     +         +         +
    .36      2         +         +     3   +         +         +
    .38      2: : : : :+: : : : :+: : : :3+: : : : :+: : : : :+
    .40      2         +         +         +3        +         +
    .42      +2        +         +         +   3     +         +
    .44      +21       +         +         +     3   +         +
    .46      +21       +         +         +       3 +         +
    .48      +2 1      +         +         +         3         +
    .50      +2  1     +         +         +         + 3       +
    .52      + 2   1   +         +         +         +  3      +
    .54      + 2    1  +         +         +         +   3     +
    .56      + 2     1 +         +         +         +     3   +
    .58      +:2: : :1+: : : : :+: : : : :+: : : : :+: : :3: :+
    .60      + 2       +1        +         +         +      3  +
    .62      + 2       + 1       +         +         +       3 +
    .64      +  2      +  1      +         +         +       3 +
    .66      +  2      +   1     +         +         +       3 +
    .68      +  2      +     1   +         +         +       3 +
    .70      +  2      +      1  +         +         +       3 +
    .72      +  2      +       1 +         +         +      3  +
    .74      +  2      +       1 +         +         +      3  +
    .76      +  2      +        1+         +         +     3   +
    .78      +: :2: : :+: : : : :1: : : : :+: : : : :+: : :3: :+
    .80      +   2     +         +1        +         +    3    +
    .82      +   2     +         +1        +         +   3     +
    .84      +   2     +         + 1       +         +  3      +
    .86      +   2     +         + 1       +         +  3      +
    .88      +   2     +         + 1       +         + 3       +
    .90      +   2     +         + 1       +         +3        +
    .92      +   2     +         +  1      +         3         +
    .94      +   2     +         +  1      +        3+         +
    .96      +   2     +         + 1       +       3 +         +
    .98      +: :2: : :+: : : : :+:1: : : :+: : :3 :+: : : : :+
SCALE 1     -.7000    -.3000     .1000     .5000     .9000    1.3000
SCALE 2     -.1000     .2000     .5000     .8000    1.1000    1.4000
SCALE 3    -1.1000    -.8000    -.5000    -.2000     .1000     .4000
```

```
MORE TIME RESPONSE RUNS (Y OR N) >N

CONTROLLER DESIGN (Y OR N) >Y

DESIGN REG/PI (Y OR N) >N

DESIGN CGT (Y OR N) >Y

MODIFY COMMAND MODEL (Y OR N) >N

KXM MATRIX

    11.61        3.312

KXU MATRIX

   -4.361

KXN MATRIX

   -1.653      -1.380

CONTROLLER EVALUATION WRT TRUTH MODEL (Y OR N) >Y

MODIFY TRUTH MODEL (Y OR N) >N

ENTER MODEL INPUT AND STEP VALUE : 1 >1 1

ENTER STATE AND IC VALUE (0/ TERMINATES):  4 >0/

2 PLOTS OF 5 VARIABLES MAY BE PRINTED AT THE TERMINAL -- SPECIFY NUMBER
FOR EACH (N1,N2) >2 0

ENTER OUTPUTS BY TYPE AND INDEX IN 2 ENTRIES--TYPES ARE
STATE : 'X'
OUTPUT : 'Y'
INPUT : 'U'
MODEL : 'M'

PLOT  1
OUTPUT  1 >Y

          >1

OUTPUT  2 >M

          >1

ENTER TIME DURATION FOR RESPONSE, IN SECONDS >1

+---------- ENTER TITLE IN GIVEN FIELD ----------+
SIMPLE DESIGN TEST, CGT/PI
```

SIMPLE DESIGN TEST, CGT/PI

```
   0.00    +         +         +    1    +  2      +         +
    .02    1         +         +         +  2      +         +
    .04    +         +         +         +  2      +     1   +
    .06    +         +         +        1+  2      +         +
    .08    +         +         +         +   2 1   +         +
    .10    +         +         +         +  12     +         +
    .12    +         +         +         +    2    +         +
    .14    +         +         +         +    2    +         +
    .16    +         +         +         +     2   +         +
    .18    +: : : : :+: : : : :+: : : : :+: : :2: :+: : : : :+
    .20    +         +         +         +      2  +         +
    .22    +         +         +         +      2  +         +
    .24    +         +         +         +       2 +         +
    .26    +         +         +         +       2 +         +
    .28    +         +         +         +        2+         +
    .30    +         +         +         +        2+         +
    .32    +         +         +         +        12         +
    .34    +         +         +         +         2         +
    .36    +         +         +         +         2         +
    .38    +: : : : :+: : : : :+: : : : :+: : : : :2: : : : :+
    .40    +         +         +         +         +2        +
    .42    +         +         +         +         +2        +
    .44    +         +         +         +         +2        +
    .46    +         +         +         +         +2        +
    .48    +         +         +         +         +2        +
    .50    +         +         +         +         +2        +
    .52    +         +         +         +         +2        +
    .54    +         +         +         +         +2        +
    .56    +         +         +         +         +21       +
    .58    +: : : : :+: : : : :+: : : : :+: : : : :+:2: : : :+
    .60    +         +         +         +         + 2       +
    .62    +         +         +         +         + 2       +
    .64    +         +         +         +         + 2       +
    .66    +         +         +         +         + 2       +
    .68    +         +         +         +         + 2       +
    .70    +         +         +         +         +21       +
    .72    +         +         +         +         +2        +
    .74    +         +         +         +         +2        +
    .76    +         +         +         +         +2        +
    .78    +: : : : :+: : : : :+: : : : :+: : : : :+2 : : : :+
    .80    +         +         +         +         +2        +
    .82    +         +         +         +         +2        +
    .84    +         +         +         +         +2        +
    .86    +         +         +         +         +2        +
    .88    +         +         +         +         +2        +
    .90    +         +         +         +         +2        +
    .92    +         +         +         +         +2        +
    .94    +         +         +         +         +2        +
    .96    +         +         +         +         +2        +
    .98    +: : : : :+: : : : :+: : : : :+: : : : :+2 : : : :+
   1.00    +         +         +         +         +2        +
 SCALE   -3.0000   -1.8000    -.6000     .6000    1.8000    3.0000
```

```
MORE TIME RESPONSE RUNS (Y OR N) >N

CONTROLLER DESIGN (Y OR N) >N

FILTER DESIGN (Y OR N) >Y

Q   MATRIX

    1.000

QN  MATRIX

    1.000

R   MATRIX

    .1000

KF  MATRIX

   6.3304E-02
   4.9510E-02
   7.4844E-02
   7.4844E-02
MODIFY TRUTH MODEL (Y OR N) >N

POLES OF FILTER  MATRIX

      -2.5838398E+00  +J(  5.4568219E+00)
      -2.5838398E+00  +J( -5.4568219E+00)
      -4.2659165E+00  +J(  0.            )
      -4.2632564E-12  +J(  0.            )
+---------- ENTER TITLE IN GIVEN FIELD ----------+
SIMPLE DESIGN TEST, KALMAN FILTER


FINAL RMS ERRORS : TRUE =   6.3143454E-02
 (STATE  1)    COMPUTED =   6.7931977E-02

FINAL RMS ERRORS : TRUE =   5.0726138E-02
 (STATE  2)    COMPUTED =   5.9937274E-02

FINAL RMS ERRORS : TRUE =   5.8594898E-02
 (STATE  3)    COMPUTED =   7.7160159E-02

FINAL RMS ERRORS : TRUE =   5.8594898E-02
 (STATE  4)    COMPUTED =   7.7160159E-02
WRITE PERFORMANCE EVALUATION DATA TO 'SAVE' FILE (Y OR N)>Y

PERFORMANCE EVALUATION DATA, NO.  1,WRITTEN TO 'SAVE  ' FILE

CONTROLLER DESIGN (Y OR N) >N
```

```
FILTER DESIGN (Y OR N) >N

END DESIGN RUNS (Y OR N) >Y

     REG/PI GAINS WRITTEN TO 'SAVE' FILE

PROGRAM EXECUTION STOP
     STOP
     065400 MAXIMUM EXECUTION FL.
      5.323 CP SECONDS EXECUTION TIME.
COMMAND- FILES

--LOCAL FILES--
 $OUTPUT    PLOT     $INPUT     CGTPIQ    SAVE
  LIST      DATA
COMMAND- CATALOG,SAVE,DATA

 INITIAL CATALOG
 RP = 008 DAYS
 CT ID=  T830287 PFN=DATA
 CT CY= 001 SN=AFIT    0000000576 WORDS.
COMMAND- RETURN,DATA

COMMAND- COPYBF,SAVE,DATA

COMMAND- RT,LIST

        LIST SENT TO PRINTER( AF ) WITH JPM.

COMMAND- RETURN,SAVE,PLOT,LIST,DATA

COMMAND- LOGOUT

CPA       6.471 SEC.        5.273 ADJ.
IO       79.214 SEC.       23.447 ADJ.
CRUS                       34.486
CONNECT TIME    0 HRS.   40 MIN.
 11/09/83  LOGGED OUT AT 22.06.48.
      <
```

B.13 CGTPIQ Output to LIST File

The first output is a heading with date and time identical to that printed at the terminal. Next the sample period of the controller is identified. A series of outputs related to the design model then follow; these are identified by a heading "DESIGN MODEL". First the matrices defining the continuous-time representation are printed, then the matrices of the discrete-time representation are printed. An additional output is the matrix Pi under a heading of "CONTROLLER SETUP".

Output relating to the design of the PI controller is identified by a heading of "REG/PI DESIGN". The quadratic weighting matrices are printed (Y, UQ, X and UM), followed by the controller gain solution GCS. Finally, the individual PI gains GC1 and GC2 are printed, as well as the intermediate product E. The proportional KP and integral KI gains are then printed. (Note, if implicit model following is requested, there are the additional outputs as discussed in Section B.8.5 of this guide.)

The truth model description is identified by the heading "TRUTH MODEL". The matrices of the continuous-time system are listed first. The eigenvalues of the matrix $\underline{A}_t$ are then printed. The matrices of the discrete-time representation are then listed.

Outputs due to the PI controller evaluation routines are identified by a heading of "CONTROLLER EVALUATION" and begin with the mapped eigenvalues of the closed-loop system with PI controller. The time-response plots are then output with each plot labeled with the title specified by the user. The plots run from 0 to 2 seconds at the controller sample period specified by the user (maximum 101 character positions).

The CGT/PI design path begins with definition of the command model, with relevant output identified by the heading "COMMAND MODEL". The continuous-time system matrices are printed, follow by the eigenvalues of the matrix $\underline{A}_m$. The discrete-time matrices are then printed. Output due to the CGT design computations is identified by the heading "CGT DESIGN". The $\underline{A}_{ij}$ matrices are then printed followed by the CGT/PI control gain matrices KXM, KXU and KXN.

The evaluation of the CGT/PI controller is also identified by the header "CONTROLLER EVALUATION". The closed-loop CGT/PI time response plots are then printed.

Output due to the Kalman filter design routine is identified by the heading "FILTER DESIGN", and includes the noise strength matrices $\underline{Q}$ and $\underline{R}$ and the Kalman filter gain matrix $\underline{K}_F$. The output of the filter evaluation routines is identified by the heading "FILTER EVALUATION". First, the mapped poles of the filter system matrix are printed. During the covariance analysis, the full error covariance matrix is printed at each time sample. Finally, the plots are printed: each plot includes the standard deviations of the "true" and filter-computed estimation error for each design model state for 50 consecutive time samples taken at the controller/filter sample period.

B.14 Summary

Appendix B has given the user of the program CGTPIQ an in-depth look at the operations of the program. This appendix was written to be a stand-alone document for normal program operation and hopefully has met that goal.

Appendix C

CGTPIQ Program Listing

C.1 Complete Listing

The following program listing includes all routines of CGTPIQ. Routines of the 'LIBRARY' object file are not listed (Ref 13).

CGTPIQ is composed of three parts: a 'MAIN' routine, an optional set of user-provided routines, and a large set of invariant routines referred to as 'CGTPIQ SUBS'. In this listing, routines 'DSND', 'DSNM', 'TRTHD', 'TRTHM', 'ACDATA', 'GUSTS', and 'TBLUP1' are the optional routines. These optional routines are used to establish the design model AFTI(S3,A2,G3) for the pitch-pointing controller, and the truth model AFTI(S4,A2,G3), both as described in Chapter VI of Floyd's thesis (Ref 9).

Included at the end of this main listing are listings of the subroutines that were changed for each of the CGT/PI formulations 2, 3 and 4 (See Section 3.3.2).

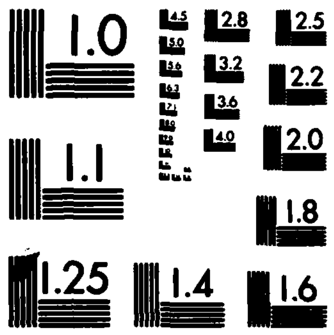
```
      PROGRAM MAIN(INPUT=64,OUTPUT=64,LIST=64,
     1 SAVE=64,DATA=64,PLOT=64,
     1 TAPE5=INPUT,TAPE6=OUTPUT,TAPE25=SAVE,TAPE50=DATA,
     2 TAPE99=PLOT,TAPE16=LIST)
      COMMON/MAIN1/NDIM,NDIM1,COM1(400)
      COMMON/MAIN2/COM2(400)
      COMMON/INOU/KIN,KOUT,KPUNCH
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/SYSMTX/NVSM,SM(2125)
      COMMON/ZMTX1/NVZM,ZM1(1225)
      COMMON/ZMTX2/ZM2(1225)
      COMMON/DSNMTX/NVDM,NODY,NOEY,DM(1750)
      COMMON/CMDMTX/NVCM,NEWCM,NODC,CM(225)
      COMMON/TRUMTX/NVTM,TM(1725)
      COMMON/CONTROL/NVCTL,CTL(900)
      COMMON/CREGPI/NVRPI,RPI(675)
      COMMON/CCGT/NVCGT,CGT(400)
      COMMON/CKF/NVFLT,FLT(690)
      COMMON/AMC/AM(100)
      COMMON/BDG/BD(75)
      NDIM=400
      NVSM=2125
      NVZM=1225
      NVDM=1750
      NVCM=225
      NVTM=1725
      NVCTL=900
      NVRPI=675
      NVCGT=400
      NVFLT=690
      KIN=5
      KSAVE=25
      KDATA=50
      KPLOT=99
      KLIST=16
      KTERM=6
      CALL CGTXQ
      STOP
C END MAIN
      END
      SUBROUTINE DSND(ND)
      DIMENSION ND(1)
      ND(1)=8
      ND(2)=2
      ND(3)=2
      ND(4)=3
      ND(5)=0
      ND(6)=1
      ND(7)=0
      RETURN
C END SUBROUTINE DSND
      END
      SUBROUTINE DSNM(A,B,EX,G,Q,C,DY,EY,H,HN,R,AN,GN,QN)
      DIMENSION A(8,8),B(8,2),C(2,8),G(8),DY(2,2),H(3,8),R(3,3)
      DATA GRAVTY,DEGTRD,PI/32.174,.01745329,3.1415927/
      CALL ACDATA(LEVEL,VT,ALT,ALPHA,ZA,ZAD,ZQ,ZU,ZDE,ZDF,
     1 PMA,PMAD,PMQ,PMU,PMDE,PMDF,XA,XAD,XQ,XU,XDE,XDF,
     2 TE,DLX,BSPAN)
 10   ALPHAR=DEGTRD*ALPHA
      U0=VT*COS(ALPHAR)
      W0=VT*SIN(ALPHAR)
      A(1,3)=1.
      A(2,1)=-GRAVTY*SIN(ALPHAR)/U0
      A(2,2)=ZA
      A(2,3)=1.+ZQ
      A(3,2)=PMA
      A(3,3)=PMQ
      A(2,7)=ZA
      A(2,8)=ZQ
      A(3,7)=PMA
      A(3,8)=PMQ
      A(2,4)=ZDE
      A(2,5)=ZDF
      A(3,4)=PMDE
      A(3,5)=PMDF
      A(4,4)=-TE
      A(5,5)=-TE
      B(4,1)=TE
      B(5,2)=TE
      CALL GUSTS(LEVEL,ALT,SLU,SLW,SIGU,SIGW)
      A(6,6)=-VT/SLW
      A(7,6)=(1.-SQRT(3.))*SIGW*SQRT(-A(6,6))/SLW
      A(7,7)=A(6,6)
      A(8,8)=-VT*PI/4./BSPAN
      A(8,6)=-A(8,8)*A(7,6)
      A(8,7)=-A(8,8)*A(7,7)
      G(6)=1.
      G(7)=SIGW*SQRT(3.*VT/SLW)/VT
      G(8)=-A(8,8)*G(7)
      Q=1.
      C(1,1)=1.
      C(2,1)=1.
      C(2,2)=-1.
      H(1,1)=1.
      H(2,2)=1.
      H(3,3)=1.
      H(2,7)=1.
      R(1,1)=4.76E-6
      R(2,2)=1.22E-5
      R(3,3)=3.22E-5
      RETURN
C END SUBROUTINE DSNM
      END
      SUBROUTINE TRTHD(ND)
      DIMENSION ND(1)
      ND(1)=9
      ND(2)=2
      ND(3)=3
      ND(4)=1
      RETURN
C END SUBROUTINE TRTHD
      END
      SUBROUTINE TRTHM(AT,BT,GT,QT,HT,RT,TDT,TNT)
      DIMENSION AT(9,9),BT(9,2),GT(9),HT(3,9),RT(3,3),TDT(8,9)
      DATA GRAVTY,DEGTRD,PI/32.174,.01745329,3.1415927/
      CALL ACDATA(LEVEL,VT,ALT,ALPHA,ZA,ZAD,ZQ,ZU,ZDE,ZDF,
     1 PMA,PMAD,PMQ,PMU,PMDE,PMDF,XA,XAD,XQ,XU,XDE,XDF,
     2 TE,DLX,BSPAN)
 10   ALPHAR=DEGTRD*ALPHA
      UO=VT*COS(ALPHAR)
      WO=VT*SIN(ALPHAR)
      RZAD=1./(1.-ZAD)
      AT(1,3)=1.
      AT(2,1)=-GRAVTY*SIN(ALPHAR)/UO
      AT(2,2)=ZA
      AT(2,3)=1.+ZQ
      AT(2,4)=ZU
      AT(3,2)=PMA
      AT(3,3)=PMQ
      AT(3,4)=PMU
      AT(4,1)=-GRAVTY*COS(ALPHAR)
      AT(4,2)=XA
      AT(4,3)=XQ-WO
      AT(4,4)=XU
      AT(2,5)=ZDE
      AT(2,6)=ZDF
      AT(3,5)=PMDE
      AT(3,6)=PMDF
      AT(4,5)=XDE
      AT(4,6)=XDF
      AT(5,5)=-TE
      AT(6,6)=-TE
      AT(2,8)=ZA
      AT(2,9)=ZQ
      AT(3,8)=PMA
      AT(3,9)=PMQ
      AT(4,8)=XA
      AT(4,9)=XQ
      CALL GUSTS(LEVEL,ALT,SLU,SLW,SIGU,SIGW)
      AT(7,7)=-VT/SLW
      AT(8,7)=(1.-SQRT(3.))*SIGW*SQRT(-AT(7,7))/SLW
      AT(8,8)=AT(7,7)
      AT(9,9)=-VT*PI/4./BSPAN
      AT(9,7)=-AT(9,9)*AT(8,7)
      AT(9,8)=-AT(9,9)*AT(8,8)
      GT(7)=1.
      GT(8)=SIGW*SQRT(3.*VT/SLW)/VT
      GT(9)=-AT(9,9)*GT(8)
      QT=1.
      DO 20 I=1,9
      AT(2,I)=AT(2,I)*RZAD
      AT(3,I)=AT(3,I)+PMAD*AT(2,I)
 20   AT(4,I)=AT(4,I)+XAD*AT(2,I)
      BT(5,1)=TE
      BT(6,2)=TE
      HT(1,1)=1.
      HT(2,2)=1.
      HT(3,3)=1.
      HT(2,8)=1.
      RT(1,1)=4.76E-6
      RT(2,2)=1.22E-5
      RT(3,3)=3.22E-5
      TDT(1,1)=1.
      TDT(2,2)=1.
      TDT(3,3)=1.
      TDT(4,5)=1.
      TDT(5,6)=1.
      TDT(6,7)=1.
      TDT(7,8)=1.
      TDT(8,9)=1.
      RETURN
C END SUBROUTINE TRTHM
      END
      SUBROUTINE ACDATA(LEVEL,VT,ALT,ALPHA,ZA,ZAD,ZQ,ZU,ZDE,ZDF,
     1 PMA,PMAD,PMQ,PMU,PMDE,PMDF,XA,XAD,XQ,XU,XDE,XDF,
     2 TE,DLX,BSPAN)
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      DATA NENTRY/1/
 5    WRITE 101
      READ*,LEVEL
      IF((LEVEL.GT.3).OR.(LEVEL.LT.1)) GO TO 5
      WRITE 102
      READ*,VT,ALT,ALPHA
      WRITE 103
      READ*,ZA,ZAD,ZQ,ZU,ZDE,ZDF
      WRITE 104
      READ*,PMA,PMAD,PMQ,PMU,PMDE,PMDF
      WRITE 105
      READ*,XA,XAD,XQ,XU,XDE,XDF
      WRITE(KLIST,101)
      WRITE(KLIST,109) LEVEL
      WRITE(KLIST,102)
      WRITE(KLIST,110) VT,ALT,ALPHA
      WRITE(KLIST,103)
      WRITE(KLIST,110) ZA,ZAD,ZQ,ZU,ZDE,ZDF
      WRITE(KLIST,104)
      WRITE(KLIST,110) PMA,PMAD,PMQ,PMU,PMDE,PMDF
      WRITE(KLIST,105)
      WRITE(KLIST,110) XA,XAD,XQ,XU,XDE,XDF
      IF(NENTRY.EQ.0) GO TO 10
      BSPAN=30.
      DLX=13.798
      TE=20.
      RETURN
 10   WRITE 106
      READ*,TE
      WRITE 107
      READ*,DLX
      WRITE 108
      READ*,BSPAN
 101  FORMAT(" ENTER TURBULENCE LEVEL (1,2,3) >")
 102  FORMAT(" ENTER TRIM VELOCITY, ALTITUDE, AND ALPHA >")
 103  FORMAT(" ENTER ZA, ZAD, ZQ, ZU, ZDE, ZDF >")
 104  FORMAT(" ENTER MA, MAD, MQ, MU, MDE, MDF >")
 105  FORMAT(" ENTER XA, XAD, XQ, XU, XDE, XDF >")
 106  FORMAT(" ENTER TIME CONSTANT FOR ELEVATOR >")
 107  FORMAT(" ENTER DISTANCE FROM CG TO ACCELEROMETER >")
 108  FORMAT(" ENTER WING SPAN >")
 109  FORMAT(6X,I1)
 110  FORMAT(6(6X1PE15.7))
      RETURN
C END SUBROUTINE ACDATA
      END
      SUBROUTINE GUSTS(LEVEL,ALT,SLU,SLW,SIGU,SIGW)
      DIMENSION ATRB1(4),ATRB2(4),ATRB3(4),SIGT1(4),SIGT2(4),SIGT3(4)
      DATA ATRB1/2000.,2750.,10000.,30000./
      DATA ATRB2/2000.,2750.,10000.,45000./
      DATA ATRB3/2000.,5000.,20000.,70000./
      DATA SIGT1/4.5,5.,5.,0./
      DATA SIGT2/8.5,10.,10.,0./
      DATA SIGT3/12.,21.,21.,0./
      DATA IT1,IT2,IT3/1,1,1/
      IF(ALT-1750.) 5,15,15
 5    IF(ALT-1000.) 8,10,10
 8    ALTT=ALT
      GO TO 12
 10   ALTT=1000.
 12   SIGW=2.5*FLOAT(LEVEL)
      SIGU=1./(.177+8.23E-4*ALTT)**.4
      SLW=ALTT
      SLU=ALTT*SIGU**3
      SIGU=SIGU*SIGW
      GO TO 100
 15   SLU=1750.
      SLW=1750.
      IF(LEVEL-2) 17,18,16
 16   CALL TBLUP1(ATRB3,SIGT3,4,IT3,ALT,SIGU)
      GO TO 19
 17   CALL TBLUP1(ATRB1,SIGT1,4,IT1,ALT,SIGU)
      GO TO 19
 18   CALL TBLUP1(ATRB2,SIGT2,4,IT2,ALT,SIGU)
 19   SIGW=SIGU
 100  RETURN
C END SUBROUTINE GUSTS
      END
      SUBROUTINE TBLUP1(X,Y,N,IXP,XP,YP)
      DIMENSION X(1),Y(1)
      IF(IXP) 15,15,1
 1    IF(IXP-N) 10,10,5
 5    IXP=N
      GO TO 18
 10   IF(XP-X(IXP)) 12,18,20
 12   IXP=IXP-1
      IF(IXP) 15,15,10
 15    IXP=1
 18    YP=Y(IXP)
       RETURN
 20    IF(IXP-N) 21,18,5
 21    IXPP1=IXP+1
 22    IF(XP-X(IXPP1)) 25,30,30
 25    YP=Y(IXP)+(XP-X(IXP))/(X(IXPP1)-X(IXP))*(Y(IXPP1)-Y(IXP))
       RETURN
 30    IXP=IXPP1
       GO TO 20
C END  SUBROUTINE TBLUP1
       END
       SUBROUTINE CGTXQ
       COMMON/MAIN1/NDIM,NDIM1,COM1(1)
       COMMON/MAIN2/COM2(1)
       COMMON/INOU/KIN,KOUT,KPUNCH
       COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
       COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
       COMMON/SYSMTX/NVSM,SM(1)
       COMMON/ZMTX1/NVZM,ZM1(1)
       COMMON/ZMTX2/ZM2(1)
       COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
       COMMON/LOCD/LAP,LGP,LPHI,LBD,LEX,LPHD,LQ,LQN,LQD,LC,LDY,LEY,LHP,LR
       COMMON/DSNMTX/NVDM,NODY,NOEY,DM(1)
       COMMON/NDIMC/NNC,NRC,NPC
       COMMON/LOCC/LPHC,LBDC,LCC,LDC
       COMMON/CMDMTX/NVCM,NEWCM,NODC,CM(1)
       COMMON/NDIMT/NNT,NRT,NMT,NWT
       COMMON/LOCT/LPHT,LBDT,LQDT,LHT,LRT,LTDT,LTNT
       COMMON/TRUMTX/NVTM,TM(1)
       COMMON/LCNTRL/LPI11,LPI12,LPI21,LPI22,LPHDL,LBDL
       COMMON/CONTROL/NVCTL,CTL(1)
       COMMON/LREGPI/LXDW,LUDW,LPHCL,LGC1,LGC2,LEL,LEE
       COMMON/CREGPI/NVRPI,RPI(1)
       COMMON/LCGT/LA11,LA13,LA21,LA23,LA12,LA22,LELA11,LELA12,LELA13
       COMMON/CCGT/NVCGT,CGT(1)
       COMMON/LKF/LEADSN,LFLTRK,LFCOV
       COMMON/CKF/NVFLT,FLT(1)
       COMMON/AMC/AM(1)
       COMMON/BDG/BD(1)
       DIMENSION LD(15),ND(15)
       DATA NPLTZM/606/
       DATA IEOI,NO/-1,1HN/
       REWIND KLIST
       WRITE(KLIST,115) DATE(DUM),TIME(DUM)
       WRITE(KTERM,115) DATE(DUM),TIME(DUM)
 115   FORMAT("1",27X,"* * * CGTPIQ * * *"/14X,
      1 "PROGRAM TO DESIGN A COMMAND GENERATOR TRACKER"/8X,
      2 "USING A REGULATOR WITH PROPORTIONAL PLUS INTEGRAL CONTROL"/13X,
      3 "BASED ON THE INTEGRAL OF THE REGULATION ERROR,"/16X,
      4 "AND A KALMAN FILTER FOR STATE ESTIMATION."/28X,
      5 "* * * CGTPIQ * * *"//11X,"DATE : ",A10//,11X,
      6 "TIME : ",A10////)
        REWIND KSAVE
      REWIND KDATA
      WRITE(KSAVE,112) IEOI,NPLTZM
      DO 10 I=1,15
 10   ND(I)=0
      DO 12 I=1,15
 12   LD(I)=1
      LFLRPI=0
      LFLCGT=0
      LFLKF=0
      LTEVAL=0
      LABORT=0
      IPI=0
      ICGT=0
      ITRU=0
      IFLTR=0
      ICODE=4
      LFAVAL=0
      LGCGT=0
      NVCOM=MINO(NDIM,NVZM)
      KOUT=KLIST
      KPUNCH=KPLOT
      IF(NVSM.GE.NPLTZM) GO TO 50
      WRITE 101,NPLTZM
      GO TO 1000
 50   WRITE 102
      READ*,TSAMP
      IF(TSAMP.LE.0.) GO TO 50
      WRITE(KLIST,103) TSAMP
 103  FORMAT("OSAMPLE PERIOD IS ",F5.3," SECONDS")
      CALL SETUP(ND,LD,ICGT,ITRU,1)
      IF(LABORT) 1000,100,1000
 100  LABORT=0
      IMPLIC=0
      WRITE 104
 104  FORMAT("OCONTROLLER DESIGN (Y OR N) >")
      READ 111,IANS
      IF(IANS.EQ.NO) GO TO 500
      LFLKF=0
      CALL PIMTX(IPI)
      IF(LABORT) 1000,125,1000
 125  WRITE 105
 105  FORMAT("ODESIGN REG/PI (Y OR N) >")
      READ 111,IANS
      IF(IANS.EQ.NO) GO TO 150
      WRITE 400
 400  FORMAT("OINCORPORATE IMPLICIT MODEL (Y OR N) >")
      READ 111,IANS
      IF(IANS.EQ.NO) GO TO 490
      IMPLIC=1
      CALL SETUP(ND,LD,ICGT,ITRU,4)
      IF(ICGT.NE.0) GO TO 460
      IMPLIC=0
      GO TO 480
 460  IF(NPD.EQ.NNC) GO TO 480
      WRITE 470
 470  FORMAT("OCOMMAND MODEL STATE DIM MUST EQUAL SYSTEM OUTPUT DIM")
      LABORT=-1
 480  IF(LABORT) 100,490,1000
 490  CALL SREGPI(IMPLIC)
      IF(LABORT) 1000,200,1000
 150  WRITE 106
 106  FORMAT("ODESIGN CGT (Y OR N) >")
      READ 111,IANS
      IF(IANS.EQ.NO) GO TO 100
      CALL SETUP(ND,LD,ICGT,ITRU,2)
      IF(ICGT) 155,100,155
 155  IF(LABORT) 100,160,1000
 160  CALL SCGT
      IF(LABORT) 100,170,1000
 170  IF(LFLCGT.LE.0) GO TO 125
 200  LABORT=0
      WRITE 107
 107  FORMAT("OCONTROLLER EVALUATION WRT TRUTH MODEL (Y OR N) >")
      READ 111,IANS
      IF(IANS.EQ.NO) GO TO 250
      CALL SETUP(ND,LD,ICGT,ITRU,3)
      IF(LABORT) 200,260,1000
 250  LTEVAL=0
 260  CALL CEVAL
      IF(LFLCGT.EQ.1) LGCGT=1
      IF(LFAVAL.EQ.0.OR.LGCGT.EQ.0) GO TO 100
 270  WRITE 600
 600  FORMAT("OWRITE PERFORMANCE EVALUATION DATA TO 'SAVE' FILE (Y OR N)
     +>")
      READ 111,IANS
      IF(IANS.EQ.NO) GO TO 100
      ICODE=ICODE+1
      CALL PFDATA(ICODE,ND)
      INUM=ICODE-4
      WRITE 605,INUM
 605  FORMAT("OPERFORMANCE EVALUATION DATA, NO. "I2,",WRITTEN TO 'SAVE
     +' FILE")
      GO TO 100
 500  LABORT=0
      WRITE 108
 108  FORMAT("OFILTER DESIGN (Y OR N) >")
      READ 111,IANS
      IF(IANS.EQ.NO) GO TO 900
      CALL FLTRK(IFLTR)
      IF(IFLTR.EQ.0) GO TO 900
      IF(LABORT) 1000,510,1000
 510  CALL SETUP(ND,LD,ICGT,ITRU,3)
      IF(LABORT) 500,525,1000
 525  CALL FEVAL
 530   IF(LABORT) 1000,540,1000
 540   LFAVAL=1
      IF(LGCGT.EQ.1) GO TO 270
      GO TO 500
 900  WRITE 109
 109  FORMAT("OEND DESIGN RUNS (Y OR N) >")
```

```
      READ 111,IANS
      IF(IANS.EQ.NO) GO TO 100
      IF(LFLRPI.EQ.0) GO TO 1000
      NPNTS=NRD*(NNPR+NND)
      ND(1)=NPNTS
      ND(2)=LGC1
      ND(3)=LGC2
      ND(4)=LEL
      CALL WFILED(4,NPNTS,ND,RPI(LGC1))
      WRITE 113
 1000 CONTINUE
      WRITE(KLIST,110)
      REWIND KSAVE
      REWIND KDATA
      REWIND KLIST
      WRITE 110
 101  FORMAT("0INSUFFICIENT MEMORY /SYSMTX/, NEED: ",I4)
 102  FORMAT("0ENTER SAMPLE PERIOD FOR DIGITAL CONTROLLER >")
 110  FORMAT("0PROGRAM EXECUTION STOP")
 111  FORMAT(A3)
 112  FORMAT(2I4)
 113  FORMAT(6X,"REG/PI GAINS WRITTEN TO 'SAVE' FILE")
      RETURN
C END SUBROUTINE CGTXQ
      END
      SUBROUTINE SETUP(ND,LD,ICGT,ITRU,ITYPE)
      DIMENSION ND(1),LD(1)
      GO TO (10,15,20,15) ITYPE
 10   CALL SDSN(ND,LD)
      RETURN
 15   CALL SCMD(ND,LD,ICGT,ITYPE)
      RETURN
 20   CALL STRTH(ND,LD,ITRU)
      RETURN
C END SUBROUTINE SETUP
      END
      SUBROUTINE SDSN(ND,LD)
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      COMMON/SYSMTX/NVSM,SM(1)
      COMMON/ZMTX1/NVZM,ZM1(1)
      COMMON/ZMTX2/ZM2(1)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      DIMENSION ND(1),LD(1)
      NSIZE=0
      CALL RSYS(SM,LD,ND,1,NSIZE)
      IF(LABORT.GT.0) RETURN
      NSIZE=NNPR
      IF(NPLD.GT.NSIZE) NSIZE=NPLD
      NSIZE=NSIZE*NSIZE
      IF(NSIZE.LE.NVCOM) GO TO 5
      WRITE 101,NSIZE
 101  FORMAT("0INSUFFICIENT MEMORY /MAIN1/,/MAIN2/,/ZMTX1/,/ZMTX2/, NEED
     1: ",I4)
      LABORT=NSIZE
      RETURN
```

AD-A138 453 COMMAND GENERATOR TRACKER SYNTHESIS METHODS USING AN LQG (LINEAR SYSTEM M..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. J P MCMILLIAN 4/4

UNCLASSIFIED DEC 83 AFIT/GE/EE/83D-46 F/G 9/2 NL

END
FILMED


1.0
4.5
5.0
5.6
6.3
7.1
8.0
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

```
 5     IF(NRD.EQ.NPD) GO TO 10
       WRITE 102
  102   FORMAT("ONUMBER OF INPUTS AND OUTPUTS MUST BE EQUAL FOR DESIGN")
       LABORT=-1
       RETURN
 10    CALL DSCRTD(LD,ZM1,ZM2)
       RETURN
C END SUBROUTINE SDSN
       END
       SUBROUTINE DSCRTD(LD,ZM1,ZM2)
       COMMON/MAIN1/NDIM,NDIM1,COM1(1)
       COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
       COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
       COMMON/SYSMTX/NVSM,SM(1)
       COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
       COMMON/LOCD/LAP,LGP,LPHI,LBD,LEX,LPHD,LQ,LQN,LQD,LC,LDY,LEY,LHP,LR
       COMMON/DSNMTX/NVDM,NODY,NOEY,DM(1)
       COMMON/LKF/LEADSN,LFLTRK,LFCOV
       COMMON/CKF/NVFLT,FLT(1)
       DIMENSION LD(1),ZM1(1),ZM2(1)
       NDIM=NPLD
       NDIM1=NDIM+1
       CALL POLES(SM,NND,1,ZM1,ZM2)
       DO 1 I=1,NND
 1     IF(ZM1(I).GT.0.) LFLCGT=-1
       CALL TFRMTX(SM,DM,NND,NND,2)
       LAP=1
       LGP=LAP+NPLD*NPLD
       IF(NWD.EQ.0) GO TO 5
       CALL TFRMTX(SM(LD(4)),DM(LGP),NND,NWD,2)
 5     IF(NDD.EQ.0) GO TO 10
       L1=LADDR(NPLD,NND+1,1)
       L2=LADDR(NPLD,1,NND+1)
       L3=LADDR(NPLD,NND+1,NND+1)
       CALL ZPART(DM(L1),NDD,NND,NPLD)
       CALL TFRMTX(SM(LD(3)),DM(L2),NND,NDD,2)
       CALL TFRMTX(SM(LD(12)),DM(L3),NDD,NDD,2)
       IF(NWD.EQ.0) GO TO 8
       L1=L1+LGP-1
       CALL ZPART(DM(L1),NDD,NWD,NPLD)
 8     L2=LADDR(NPLD,1,NWD+1)+LGP-1
       L3=LADDR(NPLD,NND+1,NWD+1)+LGP-1
       CALL ZPART(DM(L2),NND,NWDD,NPLD)
       CALL TFRMTX(SM(LD(13)),DM(L3),NDD,NWDD,2)
 10    LPHI=LGP+NPLD*NWPNWD
       LEADSN=1
       CALL NDSCRT(DM,NDIM,NT)
       CALL DSCRT(NPLD,DM,TSAMP,FLT,ZM1,NT)
       LFLTRK=LEADSN+NPLD*NPLD
       CALL TFRMTX(DM(LPHI),FLT,NND,NND,1)
       LBD=LPHI+NND*NND
       CALL TFRMTX(SM,ZM1,NND,NND,1)
       CALL FMMUL(SM,SM(LD(2)),NND,NND,NRD,DM(LBD))
       LEX=LBD+NND*NRD
       IF(NDD.EQ.0) GO TO 15
```

```
      L1=LADDR(NPLD,1,NND+1)
      CALL TFRMTX(DM(LEX),FLT(L1),NND,NDD,1)
      LPHD=LEX+NND*NDD
      L1=LADDR(NPLD,NND+1,NND+1)
      CALL TFRMTX(DM(LPHD),FLT(L1),NDD,NDD,1)
      LQ=LPHD+NDD*NDD
      GO TO 20
15    LQ=LEX
20    IF(NWD.EQ.0) GO TO 25
      CALL FTMTX(SM(LD(5)),DM(LQ),NWD,NWD)
      LQN=LQ+NWD*NWD
      GO TO 28
25    LQN=LQ
28    IF(NWDD.EQ.0) GO TO 33
      CALL FTMTX(SM(LD(14)),DM(LQN),NWDD,NWDD)
      LQD=LQN+NWDD*NWDD
      GO TO 35
33    LQD=LQN
      IF(NWPNWD.GT.0) GO TO 35
      LC=LQD
      GO TO 36
35    CALL QDSCRT(DM(LQ),DM(LQN),ZM1,ZM2)
      LC=LQD+NPLD*NPLD
36    LDY=LC+NPD*NND
      LEY=LDY+NPD*NRD
      LHP=LEY+NPD*NDD
      LR=LHP+NMD*NPLD
      L1=LR+NMD*NMD-LC
      CALL FTMTX(SM(LD(6)),DM(LC),L1,1)
      L1=LEY-1
      NODY=1
      DO 40 I=LDY,L1
      IF(DM(I).EQ.0.) GO TO 40
      NODY=0
      GO TO 45
40    CONTINUE
45    NOEY=1
      IF(NDD.LT.1) GO TO 55
      L1=LHP-1
      DO 50 I=LEY,L1
      IF(DM(I).EQ.0.) GO TO 50
      NOEY=0
      GO TO 55
50    CONTINUE
55    CALL MATLST(DM(LPHI),NND,NND,"PHI",KLIST)
      CALL MATLST(DM(LBD),NND,NRD,"BD",KLIST)
      IF(NWPNWD.GT.0) CALL MATLST(DM(LQD),NPLD,NPLD,"QD",KLIST)
      IF(NMD.GT.0) CALL MATLST(DM(LHP),NMD,NPLD,"HA",KLIST)
      IF(NDD.EQ.0) RETURN
      CALL MATLST(DM(LEX),NND,NDD,"EXD",KLIST)
      CALL MATLST(DM(LPHD),NDD,NDD,"PHN",KLIST)
      RETURN
C END SUBROUTINE DSCRTD
      END
      SUBROUTINE QDSCRT(Q,QN,ZM1,ZM2)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/LOCD/LAP,LGP,LPHI,LBD,LEX,LPHD,LQ,LQN,LQD,LC,LDY,LEY,LHP,LR
      COMMON/DSNMTX/NVDM,NODY,NOEY,DM(1)
      DIMENSION Q(1),QN(1),ZM1(1),ZM2(1)
      IF(NWD.EQ.0) GO TO 5
      CALL TFRMTX(Q,ZM1,NWD,NWD,2)
 5    IF(NWDD.EQ.0) GO TO 10
      L1=LADDR(NPLD,NWD+1,NWD+1)
      CALL TFRMTX(QN,ZM1(L1),NWDD,NWDD,2)
      IF(NWD.EQ.0) GO TO 10
      L1=LADDR(NPLD,1,NWD+1)
      CALL ZPART(ZM1(L1),NWD,NWDD,NPLD)
      L1=LADDR(NPLD,NWD+1,1)
      CALL ZPART(ZM1(L1),NWDD,NWD,NPLD)
 10   CALL MAT3(NPLD,NWPNWD,DM(LGP),ZM1,ZM2)
      CALL INTEG(NPLD,DM(LAP),ZM2,DM(LQD),TSAMP)
      RETURN
C END SUBROUTINE QDSCRT
      END
      SUBROUTINE SCMD(ND,LD,ICGT,ITYPE)
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/SYSMTX/NVSM,SM(1)
      COMMON/ZMTX1/NVZM,ZM1(1)
      COMMON/ZMTX2/ZM2(1)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/NDIMC/NNC,NRC,NPC
      COMMON/CMDMTX/NVCM,NEWCM,NODC,CM(1)
      COMMON/LREGPI/LXDW,LUDW,LPHCL,LGC1,LGC2,LEL,LEE
      COMMON/CREGPI/NVRPI,RPI(1)
      DIMENSION ND(1),LD(1)
      DATA NO/1HN/
      IF(ITYPE.EQ.4) GO TO 10
      WRITE(KLIST,110)
 110  FORMAT(////11X,5("* "),"CGT DESIGN",5(" *")////)
      IF(LFLRPI) 10,5,10
 5    WRITE 102
      READ 111,IANS
      IF(IANS.EQ.NO) GO TO 8
      CALL READFS(SM,ND,4,IERR)
      NSIZE=ND(1)
      LGC1=ND(2)
      LGC2=ND(3)
      LEL=ND(4)
      CALL FTMTX(SM,RPI(LGC1),NSIZE,1)
      IF(IERR.NE.0) RETURN
      CALL MATLST(RPI(LGC1),NRD,NND,"GC1",KLIST)
      CALL MATLST(RPI(LGC2),NRD,NRD,"GC2",KLIST)
      LFLRPI=-1
      GO TO 10
 8    IF(LFLCGT.GE.0) GO TO 9
```

```
      WRITE 103
103   FORMAT("0SYSTEM UNSTABLE - - OPEN-LOOP CGT NOT FEASIBLE")
      RETURN
9     LGC1=1
      LGC2=1
      LEL=1
      NSIZE=NRD*NND
      CALL ZPART(RPI(LGC1),1,NSIZE,1)
10    IF(ICGT.EQ.0) GO TO 12
      WRITE 108
108   FORMAT(" MODIFY COMMAND MODEL (Y OR N) >")
      READ 111,IANS
      IF(IANS.EQ.NO) RETURN
12    CALL RSYS(SM,LD,ND,2,ICGT)
      IF(LABORT.NE.0) RETURN
      NEWCM=1
      CALL POLES(SM,NNC,2,ZM1,ZM2)
      IF(NPC.EQ.NPD) GO TO 15
      WRITE 104
      LABORT=-1
      RETURN
15    CALL DSCRTC(LD,ZM1)
102   FORMAT(" READ REG/PI GAINS FROM 'DATA' FILE (Y OR N) >")
104   FORMAT("0COMMAND AND DESIGN MODEL OUTPUTS NOT EQUAL IN NUMBER")
111   FORMAT(A3)
      RETURN
C END SUBROUTINE SCMD
      END
      SUBROUTINE DSCRTC(LD,ZM1)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/SYSMTX/NVSM,SM(1)
      COMMON/NDIMC/NNC,NRC,NPC
      COMMON/LOCC/LPHC,LBDC,LCC,LDC
      COMMON/CMDMTX/NVCM,NEWCM,NODC,CM(1)
      DIMENSION LD(1),ZM1(1)
      NDIM=NNC
      NDIM1=NDIM+1
      CALL NDSCRT(SM,NDIM,NT)
      CALL DSCRT(NDIM,SM,TSAMP,CM,ZM1,NT)
      LPHC=1
      LBDC=LPHC+NNC*NNC
      CALL MMUL(ZM1,SM(LD(2)),NDIM,NDIM,NRC,CM(LBDC))
      LCC=LBDC+NNC*NRC
      LDC=LCC+NPC*NNC
      L1=LDC+NPC*NRC-LCC
      CALL FTMTX(SM(LD(3)),CM(LCC),L1,1)
      NODC=1
      L1=L1+LCC-1
      DO 10 I=LDC,L1
      IF(CM(I).EQ.0.) GO TO 10
      NODC=0
      GO TO 15
10    CONTINUE
```

```
15    CALL MATLST(CM,NNC,NNC,"PHM",KLIST)
      CALL MATLST(CM(LBDC),NNC,NRC,"BDM",KLIST)
      CALL MATLST(CM(LCC),NPC,NNC,"CM",KLIST)
      CALL MATLST(CM(LDC),NPC,NRC,"DM",KLIST)
      RETURN
C END SUBROUTINE DSCRTC
      END
      SUBROUTINE STRTH(ND,LD,ITRU)
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      COMMON/SYSMTX/NVSM,SM(1)
      COMMON/ZMTX1/NVZM,ZM1(1)
      COMMON/ZMTX2/ZM2(1)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/NDIMT/NNT,NRT,NMT,NWT
      DIMENSION ND(1),LD(1)
      DATA NO/1HN/
      IF(ITRU.EQ.0) GO TO 5
      WRITE 103
103   FORMAT(" MODIFY TRUTH MODEL (Y OR N) >")
      READ 111,IANS
111   FORMAT(A3)
      IF(IANS.EQ.NO) GO TO 20
5     CALL RSYS(SM,LD,ND,3,ITRU)
      IF(LABORT.GT.0) RETURN
      NSIZE=NNT*NNT
      IF(NSIZE.LE.NVCOM) GO TO 8
      WRITE 101,NSIZE
101   FORMAT("0INSUFFICIENT MEMORY /MAIN1/,/MAIN2/,/ZMTX1/,/ZMTX2/, NEED
     1: ",I2)
      LABORT=NSIZE
      RETURN
8     IF((NRT.EQ.NRD).AND.(NMT.EQ.NMD)) GO TO 10
      WRITE 102
102   FORMAT("0INPUTS AND MEASUREMENTS MUST BE EQUAL IN NUMBER FOR DESIG
     1N AND TRUTH MODELS")
      LABORT=-1
      RETURN
10    CALL POLES(SM,NNT,3,ZM1,ZM2)
      CALL DSCRTT(LD,ZM1)
20    LTEVAL=1
      RETURN
C END SUBROUTINE STRTH
      END
      SUBROUTINE DSCRTT(LD,ZM1)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      COMMON/SYSMTX/NVSM,SM(1)
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/NDIMT/NNT,NRT,NMT,NWT
      COMMON/LOCT/LPHT,LBDT,LQDT,LHT,LRT,LTDT,LTNT
      COMMON/TRUMTX/NVTM,TM(1)
      DIMENSION LD(1),ZM1(1)
      NDIM=NNT
      NDIM1=NDIM+1
      CALL NDSCRT(SM,NDIM,NT)
      CALL DSCRT(NDIM,SM,TSAMP,TM,ZM1,NT)
      LPHT=1
      LBDT=LPHT+NNT*NNT
      CALL MMUL(ZM1,SM(LD(2)),NDIM,NDIM,NRT,TM(LBDT))
      LQDT=LBDT+NNT*NRT
      IF(NWT.GT.0) GO TO 10
      LHT=LQDT
      GO TO 15
 10   CALL MAT3(NDIM,NWT,SM(LD(3)),SM(LD(4)),ZM1)
      CALL INTEG(NDIM,SM,ZM1,TM(LQDT),TSAMP)
      LHT=LQDT+NNT*NNT
 15   LRT=LHT+NMT*NNT
      LTDT=LRT+NMT*NMT
      LTNT=LTDT+NND*NNT
      L1=LTNT+NDD*NNT-LHT
      CALL FTMTX(SM(LD(5)),TM(LHT),L1,1)
      CALL MATLST(TM,NNT,NNT,"PHT",KLIST)
      CALL MATLST(TM(LBDT),NNT,NRT,"BDT",KLIST)
      IF(NWT.GT.0) CALL MATLST(TM(LQDT),NNT,NNT,"QDT",KLIST)
      RETURN
C END SUBROUTINE DSCRTT
      END
      SUBROUTINE PIMTX(IPI)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/ZMTX1/NVZM,ZM1(1)
      COMMON/ZMTX2/ZM2(1)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/LOCD/LAP,LGP,LPHI,LBD,LEX,LPHD,LQ,LQN,LQD,LC,LDY,LEY,LHP,LR
      COMMON/DSNMTX/NVDM,NODY,NOEY,DM(1)
      COMMON/LCNTRL/LPI11,LPI12,LPI21,LPI22,LPHDL,LBDL
      COMMON/CONTROL/NVCTL,CTL(1)
      IF(IPI.EQ.1) RETURN
      WRITE(KLIST,110)
 110  FORMAT(////11X,5("* "),"CONTROLLER SET-UP",5(" *")////)
      NDIM=NNPR
      NSIZE=NDIM*(2*NDIM+NPD)
      IF(NSIZE.LE.NVCTL) GO TO 10
      WRITE 101,NSIZE
 101  FORMAT("0INSUFFICIENT MEMORY /CONTROL/, NEED: ",I4)
      LABORT=NSIZE
      RETURN
 10   NDIM1=NDIM+1
      LPI11=1
      LPI12=LPI11+NND*NND
      LPI21=LPI12+NND*NRD
      LPI22=LPI21+NPD*NND
      LPHDL=LPI22+NPD*NRD
      CALL TFRMTX(DM(LPHI),ZM1,NND,NND,2)
      CALL SUBI(ZM1,NND,NDIM)
      L2=LADDR(NDIM,1,NND+1)
      CALL TFRMTX(DM(LBD),ZM1(L2),NND,NRD,2)
      L3=LADDR(NDIM,NND+1,1)
      CALL TFRMTX(DM(LC),ZM1(L3),NPD,NND,2)
      L4=LADDR(NDIM,NND+1,NND+1)
      CALL TFRMTX(DM(LDY),ZM1(L4),NPD,NRD,2)
      CALL GMINV(NDIM,NDIM,ZM1,ZM2,MR,1)
      IF(MR.EQ.NDIM) GO TO 15
      WRITE 102
      WRITE(KLIST,102)
  102 FORMAT("0PI MATRIX IS RANK DEFECTIVE")
   15 CALL MATLST(ZM2,NNPR,NNPR,"PI",KLIST)
      CALL TFRMTX(CTL(LPI11),ZM2,NND,NND,1)
      CALL TFRMTX(CTL(LPI12),ZM2(L2),NND,NRD,1)
      CALL TFRMTX(CTL(LPI21),ZM2(L3),NPD,NND,1)
      CALL TFRMTX(CTL(LPI22),ZM2(L4),NPD,NRD,1)
      CALL CDIF
      IPI=1
      RETURN
C END SUBROUTINE PIMTX
      END
      SUBROUTINE CDIF
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/LOCD/LAP,LGP,LPHI,LBD,LEX,LPHD,LQ,LQN,LQD,LC,LDY,LEY,LHP,LR
      COMMON/DSNMTX/NVDM,NODY,NOEY,DM(1)
      COMMON/LCNTRL/LPI11,LPI12,LPI21,LPI22,LPHDL,LBDL
      COMMON/CONTROL/NVCTL,CTL(1)
      CALL TFRMTX(DM(LPHI),CTL(LPHDL),NND,NND,2)
      L1=LADDR(NDIM,1,NND+1)+LPHDL-1
      CALL ZPART(CTL(L1),NND,NPD,NDIM)
      L1=LADDR(NDIM,NND+1,1)+LPHDL-1
      CALL TFRMTX(DM(LC),CTL(L1),NPD,NND,2)
      L1=LADDR(NDIM,NND+1,NND+1)+LPHDL-1
      CALL IDNT(NRD,CTL(L1),1.)
      LBDL=LPHDL+NDIM*NDIM
      CALL TFRMTX(DM(LBD),CTL(LBDL),NND,NRD,2)
      L1=LADDR(NDIM,NND+1,1)+LBDL-1
      CALL TFRMTX(DM(LDY),CTL(L1),NPD,NRD,2)
      RETURN
C END SUBROUTINE CDIF
      END
      SUBROUTINE SREGPI(IMPLIC)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/SYSMTX/NVSM,SM(1)
      COMMON/ZMTX1/NVZM,ZM1(1)
      COMMON/ZMTX2/ZM2(1)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/LCNTRL/LPI11,LPI12,LPI21,LPI22,LPHDL,LBDL
      COMMON/CONTROL/NVCTL,CTL(1)
      COMMON/LREGPI/LXDW,LUDW,LPHCL,LGC1,LGC2,LEL,LEE
      COMMON/CREGPI/NVRPI,RPI(1)
      COMMON/LOCD/LAP,LGP,LPHI,LBD,LEX,LPHD,LQ,LQN,LQD,LC,LDY,LEY,LHP,LR
      COMMON/DSNMTX/NVDM,NODY,NOEY,DM(1)
      WRITE(KLIST,110)
  110 FORMAT(////11X,5("* "),"REG/PI DESIGN",5(" *")////)
```

```
      NSIZE=NRD*(5*NRD+2*NND)+NNPR*NNPR
      IF(NSIZE.LE.NVRPI) GO TO 5
      WRITE 101,NSIZE
101   FORMAT("0INSUFFICIENT MEMORY /CREGPI/, NEED: ",I4)
      GO TO 8
5     NSIZE=NNPR*(3*NNPR+NRD)
      IF(NSIZE.LE.NVSM) GO TO 10
      WRITE 102,NSIZE
102   FORMAT("0INSUFFICIENT MEMORY /SYSMTX/, NEED: ",I4)
8     LABORT=NSIZE
      RETURN
10    LX=1
      LU=LX+NNPR*NNPR
      CALL WXUS(SM(LX),SM(LU),COM1,ZM1,ZM2,IMPLIC)
      LUIST=LU+NNPR*NRD
      LPHP=LUIST+NNPR*NNPR
      CALL PXUP(CTL(LPHDL),CTL(LBDL),SM(LX),SM(LU),COM1,ZM2,
     1 SM(LUIST),SM(LPHP),SM(LX),ZM1)
      CALL DRIC(NDIM,SM(LPHP),ZM2,SM(LX),ZM1,RPI(LPHCL))
      CALL GCSTAR(SM(LPHP),CTL(LBDL),SM(LU),ZM1,SM(LUIST),SM(LX),ZM2)
      CALL TFRMTX(RPI(LGC1),SM(LX),NRD,NND,1)
      L3=LADDR(NDIM,1,NND+1)+LX-1
      CALL TFRMTX(RPI(LGC2),SM(L3),NRD,NRD,1)
      L1=LADDR(NNPR,NND+1,NND+1)
      L2=LADDR(NNPR,1,NND+1)
      CALL GMINV(NRD,NRD,ZM1(L1),ZM1(L2),MR,1)
      L1=LADDR(NNPR,NND+1,1)
      CALL MMUL(ZM1(L2),ZM1(L1),NRD,NRD,NND,ZM1)
      CALL MMUL(SM(L3),ZM1,NRD,NRD,NND,ZM1(L1))
      CALL MADD1(NRD,NND,SM(LX),ZM1(L1),ZM1,-1.)
      CALL TFRMTX(RPI(LEL),ZM1,NRD,NND,1)
      CALL FMMUL(RPI(LEL),CTL(LPI12),NRD,NND,NRD,RPI(LEE))
      CALL FMADD(RPI(LEE),CTL(LPI22),NRD,NRD,RPI(LEE))
      CALL MATLST(RPI(LGC1),NRD,NND,"GC1",KLIST)
      CALL MATLST(RPI(LGC1),NRD,NND,"GC1",KTERM)
      CALL MATLST(RPI(LGC2),NRD,NRD,"GC2",KLIST)
      CALL MATLST(RPI(LGC2),NRD,NRD,"GC2",KTERM)
      CALL MATLST(RPI(LEE),NRD,NRD,"E",KLIST)
      IF(NODY.EQ.1)GO TO 15
      CALL FMMUL(RPI(LEE),DM(LDY),NRD,NRD,NRD,ZM1)
      CALL SUBI(ZM1,NRD,NRD)
      NR2=NRD*NRD
      DO 12 I=1,NR2
12    ZM1(I)=-ZM1(I)
      NDIM=NRD
      NDIM1=NDIM+1
      CALL GMINV(NRD,NRD,ZM1,ZM2,MR,1)
      CALL FMMUL(ZM2,RPI(LEE),NRD,NRD,NRD,ZM1)
      CALL MATLST(ZM1,NRD,NRD,"KP",KLIST)
      CALL FMMUL(ZM2,RPI(LGC2),NRD,NRD,NRD,ZM1)
      CALL MATLST(ZM1,NRD,NRD,"KI",KLIST)
      GO TO 20
15    WRITE(KLIST,103)
```

```
  103 FORMAT("0KP = E  AND  KI = GC2")
  20  CONTINUE
      LFLRPI=1
      LFLCGT=0
      RETURN
C END SUBROUTINE SREGPI
      END
      SUBROUTINE WXUS(X,U,S,ZM1,ZM2,IMPLIC)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/MAIN2/COM2(1)
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/SYSMTX/NVSM,SM(1)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/LOCD/LAP,LGP,LPHI,LBD,LEX,LPHD,LQ,LQN,LQD,LC,LDY,LEY,LHP,LR
      COMMON/DSNMTX/NVDM,NODY,NOEY,DM(1)
      COMMON/LCNTRL/LPI11,LPI12,LPI21,LPI22,LPHDL,LBDL
      COMMON/CONTROL/NVCTL,CTL(1)
      COMMON/LREGPI/LXDW,LUDW,LPHCL,LGC1,LGC2,LEL,LEE
      COMMON/CREGPI/NVRPI,RPI(1)
      DIMENSION X(1),U(1),S(1),ZM1(1),ZM2(1)
      DATA NO/1HN/
      IF(IMPLIC.EQ.0) GO TO 2
      CALL IMPLEX(ZM1)
  2   IF(LFLRPI) 5,5,10
  5   LXDW=1
      LUDW=LXDW+2*NRD*NRD
      LPHCL=LUDW+NRD*NRD
      LGC1=LPHCL+NNPR*NNPR
      LGC2=LGC1+NRD*NND
      LEL=LGC2+NRD*NRD
      LEE=LEL+NRD*NND
      L1=LPHCL-1
      CALL ZPART(RPI,1,L1,1)
  10  LUX=NRD*NRD+1
      WRITE 101,NPD
  101 FORMAT(" ENTER WEIGHTS ON OUTPUT DEVIATIONS: ",I2)
      CALL RQWGTS(RPI,NPD,0)
      WRITE 102,NRD
  102 FORMAT(" ENTER WEIGHTS ON INTEGRAL OF REGULATION ERROR: ",I2)
      CALL RQWGTS(RPI(LUX),NRD,1)
      WRITE 103,NRD
  103 FORMAT(" ENTER WEIGHTS ON CONTROL MAGNITUDES: ",I2)
      CALL RQWGTS(RPI(LUDW),NRD,1)
      CALL MATLST(RPI,NPD,NPD,"Y",KLIST)
      CALL DVCTOR(NPD,RPI,ZM1)
      CALL MATLST(ZM1,NPD,1,"Y",KTERM)
      CALL DVCTOR(NRD,RPI(LUX),ZM1)
      CALL MATLST(ZM1,NRD,1,"UQ",KTERM)
      CALL MATLST(RPI(LUX),NRD,NRD,"UQ",KLIST)
      NDIM=NNPR
      NDIM1=NDIM+1
      CALL FORMXU(RPI,RPI(LUX),RPI(LUDW),DM(LC),DM(LDY),ZM2,ZM1,COM1)
      WRITE(KTERM,104)
  104 FORMAT("0MODIFY ELEMENTS OF 'X' MATRIX (Y OR N) >")
```

```
      READ 111,IANS
 111  FORMAT(A3)
      IF(IANS.EQ.NO) GO TO 20
      WRITE(KTERM,105)
 105  FORMAT(" LIST 'X' MATRIX TO TERMINAL (Y OR N) >")
      READ 111,IANS
      IF(IANS.EQ.NO) GO TO 12
      CALL MATLST(ZM2,NNPR,NNPR,"X",KTERM)
 12   CALL ZMATIN(ZM2,NNPR,NNPR,-1)
 20   CALL MATLST(ZM2,NNPR,NNPR,"X",KLIST)
      CALL MATLST(RPI(LUDW),NRD,NRD,"UM",KLIST)
      CALL DVCTOR(NRD,RPI(LUDW),ZM1)
      CALL MATLST(ZM1,NRD,1,"UM",KTERM)
      IF(IMPLIC.EQ.0) GO TO 260
         CALL MODXU(ZM2,RPI(LUDW))
 260  CONTINUE
      T1=.25*TSAMP
      CALL MSCALE(ZM1,ZM2,NDIM,NDIM,T1)
      CALL DIAG(NDIM,COM1,CTL(LPHDL),1.,1.)
      CALL MAT3A(NDIM,NDIM,COM1,ZM1,X)
      CALL MAT3A(NRD,NDIM,CTL(LBDL),ZM1,U)
      CALL MAT4A(COM1,ZM1,NDIM,NDIM,NDIM,ZM2)
      CALL MMUL(ZM2,CTL(LBDL),NDIM,NDIM,NRD,S)
      T1=.5*TSAMP
      CALL MSCALE(COM2,COM2,NNPR,NRD,T1)
      CALL MAT4A(COM1,COM2,NDIM,NDIM,NRD,ZM1)
      CALL MADD1(NDIM,NRD,ZM1,S,S,1.)
      CALL FTMUL(CTL(LBDL),COM2,NDIM,NRD,NRD,COM1)
      NRD2=NRD*NRD
      DO 25 I=1,NRD
         K=(I-1)*NRD
         JK=K+LUDW
         DO 25 J=I,NRD2,NRD
         K=K+1
         COM2(K)=COM1(K)+COM1(J)+RPI(JK)*TSAMP
 25   JK=JK+1
      CALL TFRMTX(COM2,ZM1,NRD,NRD,2)
      CALL MADD1(NRD,NRD,U,ZM1,U,1.)
      RETURN
C END  SUBROUTINE WXUS
      END
      SUBROUTINE FORMXU(QY,RY,RU,C,D,X,Z1,Z2)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/MAIN2/COM2(1)
      DIMENSION QY(1),RY(1),C(1),D(1),X(1),Z1(1),Z2(1)
      CALL ZPART(X,NNPR,NNPR,NNPR)
      CALL FTMUL(C,QY,NPD,NND,NPD,Z1)
      CALL FMMUL(Z1,C,NND,NPD,NND,Z2)
      CALL TFRMTX(Z2,X,NND,NND,2)
      L1=LADDR(NNPR,NND+1,NND+1)
      CALL TFRMTX(RY,X(L1) NRD,NRD,2)
      L2=LADDR(N[illegible].NND[illegible],[illegible])
      CALL ZPART([illegible](L[illegible],,NRD,NRD,NNPR)
      IF(NODY.EQ.0) GO TO 5
      CALL ZPART(COM2,NND,NRD,NNPR)
      RETURN
    5 CALL FMMUL(Z1,D,NND,NPD,NRD,Z2)
      CALL TFRMTX(Z2,COM2,NND,NRD,2)
      CALL FMMUL(QY,D,NPD,NPD,NRD,Z1)
      CALL FTMUL(D,Z1,NPD,NRD,NRD,Z2)
      CALL FMADD(Z2,RU,NRD,NRD,RU)
      RETURN
C END SUBROUTINE FORMXU
      END
      SUBROUTINE PXUP(PHIDL,BDEL,X,U,S,BUIBT,UIST,PHIP,XP,ZM1)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      DIMENSION PHIDL(1),BDEL(1),X(1),U(1),S(1),BUIBT(1),UIST(1),
     1 PHIP(1),XP(1),ZM1(1)
      CALL EQUATE(PHIP,U,NRD,NRD)
      CALL GMINV(NRD,NRD,PHIP,ZM1,MR,1)
      CALL MAT3(NDIM,NRD,BDEL,ZM1,BUIBT)
      CALL MAT5(ZM1,S,NRD,NRD,NDIM,UIST)
      CALL MMUL(BDEL,UIST,NDIM,NRD,NDIM,ZM1)
      CALL MADD1(NDIM,NDIM,PHIDL,ZM1,PHIP,-1.)
      CALL MMUL(S,UIST,NDIM,NRD,NDIM,ZM1)
      CALL MADD1(NDIM,NDIM,X,ZM1,XP,-1.)
      RETURN
C END SUBROUTINE PXUP
      END
      SUBROUTINE GCSTAR(PHIP,BDEL,U,RK,UIST,GCS,ZM1)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      DIMENSION PHIP(1),BDEL(1),U(1),RK(1),UIST(1),GCS(1),ZM1(1)
      CALL MAT3A(NRD,NDIM,BDEL,RK,ZM1)
      CALL MADD1(NRD,NRD,ZM1,U,ZM1,1.)
      CALL GMINV(NRD,NRD,ZM1,U,MR,1)
      CALL MAT5(U,BDEL,NRD,NRD,NDIM,ZM1)
      CALL MAT1(ZM1,RK,NRD,NDIM,NDIM,GCS)
      CALL MMUL(GCS,PHIP,NRD,NDIM,NDIM,ZM1)
      CALL MADD1(NRD,NDIM,ZM1,UIST,GCS,1.)
      WRITE(KLIST,101)
  101 FORMAT("OREG/PI GAIN MATRIX--GCS"/)
      CALL MATIO(GCS,NRD,NDIM,3)
      RETURN
C END SUBROUTINE GCSTAR
      END
      SUBROUTINE SCGT
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/ZMTX1/NVZM,ZM1(1)
      COMMON/ZMTX2/ZM2(1)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/NDIMC/NNC,NRC,NPC
      COMMON/CMDMTX/NVCM,NEWCM,NODC,CM(1)
      COMMON/LREGPI/LXDW,LUDW,LPHCL,LGC1,LGC2,LEL,LEE
      COMMON/CREGPI/NVRPI,RPI(1)
      COMMON/LCGT/LA11,LA13,LA21,LA23,LA12,LA22,LELA11,LELA12,LELA13
      COMMON/CCGT/NVCGT,CGT(1)
      IF(NEWCM) 20,20,15
15    NSIZE=(NND+2*NPD)*(NNC+NRC+NDD)
      IF(NSIZE.LE.NVCGT) GO TO 16
      WRITE 106,NSIZE
      LABORT=NSIZE
      RETURN
16    IF(NND.GE.NNC) GO TO 17
      WRITE 107
      GO TO 18
17    IF(NND.GE.NDD) GO TO 19
      WRITE 108
18    LABORT=-1
      RETURN
19    NEWCM=0
      LA11=1
      LA13=LA11+NND*NNC
      LA21=LA13+NND*NDD
      LA23=LA21+NPD*NNC
      LA12=LA23+NPD*NDD
      LA22=LA12+NND*NRC
      LELA11=LA22+NPD*NRC
      LELA12=LELA11+NPD*NNC
      LELA13=LELA12+NPD*NRC
      CALL CGTA(CGT(LA11),CGT(LA13),CGT(LA21),CGT(LA23),CGT(LA12),
     1 CGT(LA22),ZM1,ZM2)
20    CALL CGTKX(CGT(LA11),CGT(LA13),CGT(LA21),CGT(LA23),CGT(LA12),
     1 CGT(LA22),CGT(LELA11),CGT(LELA12),CGT(LELA13),RPI(LEL))
      LFLCGT=1
106   FORMAT("0INSUFFICIENT MEMORY /CCGT/, NEED: ",I4)
107   FORMAT("0FEWER DESIGN MODEL THAN COMMAND MODEL STATES")
108   FORMAT("FEWER DESIGN MODEL THAN DISTURBANCE MODEL STATES")
      RETURN
C END SUBROUTINE SCGT
      END
      SUBROUTINE CGTA(A11,A13,A21,A23,A12,A22,ZM1,ZM2)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/SYSMTX/NVSM,SM(1)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/LOCD/LAP,LGP,LPHI,LBD,LEX,LPHD,LQ,LQN,LQD,LC,LDY,LEY,LHP,LR
      COMMON/DSNMTX/NVDM,NODY,NOEY,DM(1)
      COMMON/NDIMC/NNC,NRC,NPC
      COMMON/LOCC/LPHC,LBDC,LCC,LDC
      COMMON/CMDMTX/NVCM,NEWCM,NODC,CM(1)
      COMMON/LCNTRL/LPI11,LPI12,LPI21,LPI22,LPHDL,LBDL
      COMMON/CONTROL/NVCTL,CTL(1)
      DIMENSION A11(1),A13(1),A21(1),A23(1),A12(1),A22(1),ZM1(1),ZM2(1)
      NDIM=NND
      NDIM1=NDIM+1
      CALL TFRMTX(CM,ZM1,NNC,NNC,2)
      CALL SUBI(ZM1,NNC,NDIM)
      CALL FMMUL(CTL(LPI12),CM(LCC),NND,NRD,NNC,ZM2)
      CALL MSCALE(ZM2,ZM2,NND,NNC,-1.)
      NB=MAX0(NDD,NNC)
      L2=1+NND*NND
      L3=L2+NND*NB
      L4=L3+NND*NB
      L5=L4+NND*NND
      L6=L5+NND*NND
      NSIZE=L6+NPD*NNC-1
      IF(NSIZE.LE.NVSM) GO TO 1
      WRITE 102,NSIZE
102   FORMAT("OINSUFFICIENT MEMORY /SYSMTX/, NEED: ",I4)
      LABORT=NSIZE
      RETURN
1     CALL AXBMXC(CTL(LPI11),NND,ZM1,NNC,ZM2,A11,SM,
     1 SM(L2),SM(L3),SM(L4),SM(L5))
      CALL MMUL(A11,ZM1,NND,NNC,NNC,ZM2)
      CALL FMMUL(CTL(LPI21),ZM2,NPD,NND,NNC,A21)
      CALL FMMUL(CTL(LPI22),CM(LCC),NPD,NRD,NNC,SM(L6))
      CALL FMMUL(A11,CM(LBDC),NND,NNC,NRC,SM)
      CALL FMMUL(CTL(LPI11),SM,NND,NND,NRC,A12)
      CALL FMMUL(CTL(LPI21),SM,NPD,NND,NRC,A22)
      IF(NODC.EQ.1) GO TO 2
      CALL FMMUL(CTL(LPI12),CM(LDC),NND,NRD,NRC,SM(L2))
      CALL FMADD(A12,SM(L2),NND,NRC,A12)
      CALL FMMUL(CTL(LPI22),CM(LDC),NPD,NRD,NRC,SM(L2))
      CALL FMADD(A22,SM(L2),NPD,NRC,A22)
2     IF(NDD.EQ.0) GO TO 15
      CALL MMUL(CTL(LPI11),DM(LEX),NND,NND,NDD,ZM2)
      IF(NOEY.EQ.1) GO TO 5
      CALL FMMUL(CTL(LPI12),DM(LEY),NND,NRD,NDD,ZM1)
      CALL MADD1(NND,NDD,ZM1,ZM2,ZM2,1.)
5     CALL TFRMTX(DM(LPHD),ZM1,NDD,NDD,2)
      CALL SUBI(ZM1,NDD,NDIM)
      CALL AXBMXC(CTL(LPI11),NND,ZM1,NDD,ZM2,A13,SM,
     1 SM(L2),SM(L3),SM(L4),SM(L5))
      CALL MMUL(A13,ZM1,NND,NDD,NDD,ZM2)
      CALL MADD1(NND,NDD,ZM2,DM(LEX),ZM2,-1.)
      CALL FMMUL(CTL(LPI21),ZM2,NPD,NND,NDD,A23)
15    NDIM=NPD
      NDIM1=NDIM+1
      CALL MADD1(NPD,NNC,A21,SM(L6),A21,1.)

      IF(NOEY.EQ.1) GO TO 20
      CALL MMUL(CTL(LPI22),DM(LEY),NPD,NRD,NDD,ZM1)
      CALL MADD1(NPD,NDD,A23,ZM1,A23,-1.)
20    CALL MATLST(A11,NND,NNC,"A11",KLIST)
      CALL MATLST(A21,NPD,NNC,"A21",KLIST)
      CALL MATLST(A12,NND,NRC,"A12",KLIST)
      CALL MATLST(A22,NPD,NRC,"A22",KLIST)
      IF(NDD.GT.0) GO TO 25
      WRITE(KLIST,101)
101   FORMAT("OMATRICES A13 AND A23 ARE ZERO")
      RETURN
25    CALL MATLST(A13,NND,NDD,"A13",KLIST)
      CALL MATLST(A23,NPD,NDD,"A23",KLIST)
      RETURN
C END SUBROUTINE CGTA
      END
      SUBROUTINE AXBMXC(A,NA,B,NB,C,X,AU,BU,R,Z1,Z2)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      DIMENSION A(1),B(1),C(1),X(1),AU(1),BU(1),R(1),Z1(1),Z2(1)
      DATA EMAX,ITMAX/1.E-6,3/
      CALL TRANS1(NA,A,Z1)
      CALL EIGEN(NA,Z1,Z2,Z2(NDIM1),AU,1)
      CALL TRANS1(NA,COM1,Z1)
      CALL EIGEN(NB,B,Z2,Z2(NDIM1),BU,1)
      CALL EQUATE(R,C,NA,NB)
      IT=0
 10   CALL MAT4A(AU,R,NA,NA,NB,Z2)
      CALL MAT1(Z2,BU,NA,NB,NB,R)
      CALL SLVSHR(Z1,NA,COM1,NB,R,NDIM)
      CALL MAT4(R,BU,NA,NB,NB,Z2)
      CALL MAT1(AU,Z2,NA,NA,NB,R)
      IF(IT.GT.0) GO TO 15
      CALL EQUATE(X,R,NA,NB)
      GO TO 30
 15   CALL MADD1(NA,NB,X,R,X,1.)
      CALL ENORM(R,NA,NB,EN)
      IF(EN.LE.EMAX) RETURN
      IF(IT.LT.ITMAX) GO TO 30
      WRITE(KLIST,101) EN
      WRITE(KTERM,101) EN
 101  FORMAT("OSOLUTION ERROR FOR 'A'(CGT) AFTER 3 ITERATIONS = ",1PE15.
     17)
      RETURN
 30   CALL MAT1(A,X,NA,NA,NB,Z2)
      CALL MAT1(Z2,B,NA,NB,NB,R)
      CALL MADD1(NA,NB,X,R,R,-1.)
      CALL MADD1(NA,NB,R,C,R,1.)
      IT=IT+1
      GO TO 10
C END SUBROUTINE AXBMXC
      END
      SUBROUTINE SLVSHR(A,NA,B,NB,C,ND)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/INOU/KIN,KOUT,KPUNCH
      DIMENSION A(ND,1),B(ND,1),C(ND,1),V(16),W(4)
      L=1
 5    LM1=L-1
      DL=1
      IF(L.EQ.NB) GO TO 8
      IF(B(L+1,L).NE.0.) DL=2
 8    LL=LM1+DL
      K=1
 10   KM1=K-1
      DK=1
      IF(K.EQ.NA) GO TO 12
      IF(A(K,K+1).NE.0.) DK=2
 12   KK=KM1+DK
      AKK=A(K,K)
      BLL=B(L,L)
      IF(DL.EQ.2) GO TO 35
      IF(DK.EQ.2) GO TO 20
      IF(L.EQ.1) GO TO 13
      C(K,L)=C(K,L)-AKK*DOT3(LM1,C(K,1),B(1,L))
13    IF(K.EQ.1) GO TO 18
      DO 15 I=1,KM1
15    C(K,L)=C(K,L)-A(K,I)*DOT3(L,C(I,1),B(1,L))
18    V(1)=AKK*BLL-1.
      IF(V(1).EQ.0.) GO TO 99
      C(K,L)=C(K,L)/V(1)
      GO TO 95
20    IF(L.EQ.1) GO TO 22
      I1=K
      I2=KK
      I3=LM1
      GO TO 24
22    IF(K.EQ.1) GO TO 30
      I1=1
      I2=KM1
      I3=L
24    DO 28 I=I1,I2
      V(1)=DOT3(I3,C(I,1),B(1,L))
      C(K,L)=C(K,L)-A(K,I)*V(1)
28    C(KK,L)=C(KK,L)-A(KK,I)*V(1)
      IF(I1.EQ.K) GO TO 22
30    V(1)=AKK*BLL-1.
      V(2)=A(KK,K)*BLL
      V(3)=A(K,KK)*BLL
      V(4)=A(KK,KK)*BLL-1.
      V(5)=1./(V(1)*V(4)-V(2)*V(3))
      V(6)=V(5)*(C(K,L)*V(4)-V(3)*C(KK,L))
      C(KK,L)=V(5)*(V(1)*C(KK,L)-V(2)*C(K,L))
      C(K,L)=V(6)
      GO TO 95
35    IF(DK.EQ.2) GO TO 50
      IF(L.EQ.1) GO TO 38
      I1=K
      I2=K
      I3=LM1
      GO TO 40
38    IF(K.EQ.1) GO TO 45
      I1=1
      I2=KM1
      I3=LL
40    DO 42 I=I1,I2
      C(K,L)=C(K,L)-A(K,I)*DOT3(I3,C(I,1),B(1,L))
42    C(K,LL)=C(K,LL)-A(K,I)*DOT3(I3,C(I,1),B(1,LL))
      IF(I1.EQ.K) GO TO 38
45    V(1)=AKK*BLL-1.
      V(2)=AKK*B(L,LL)
      V(3)=AKK*B(LL,L)
      V(4)=AKK*B(LL,LL)-1.
48    V(5)=1./(V(1)*V(4)-V(2)*V(3))
      V(6)=V(5)*(C(K,L)*V(4)-V(3)*C(K,LL))
      C(K,LL)=V(5)*(V(1)*C(K,LL)-V(2)*C(K,L))
      C(K,L)=V(6)
      GO TO 95
 50   IF(L.EQ.1) GO TO 55
      V(1)=DOT3(LM1,C(K,1),B(1,L))
      V(2)=DOT3(LM1,C(KK,1),B(1,L))
      V(3)=DOT3(LM1,C(K,1),B(1,LL))
      V(4)=DOT3(LM1,C(KK,1),B(1,LL))
      C(K,L)=C(K,L)-AKK*V(1)-A(K,KK)*V(2)
      C(KK,L)=C(KK,L)-A(KK,K)*V(1)-A(KK,KK)*V(2)
      C(K,LL)=C(K,LL)-AKK*V(3)-A(K,KK)*V(4)
      C(KK,LL)=C(KK,LL)-A(KK,K)*V(3)-A(KK,KK)*V(4)
 55   IF(K.EQ.1) GO TO 65
      DO 60 I=1,KM1
      V(1)=DOT3(LL,C(I,1),B(1,L))
      V(2)=DOT3(LL,C(I,1),B(1,LL))
      C(K,L)=C(K,L)-A(K,I)*V(1)
      C(KK,L)=C(KK,L)-A(KK,I)*V(1)
      C(K,LL)=C(K,LL)-A(K,I)*V(2)
 60   C(KK,LL)=C(KK,LL)-A(KK,I)*V(2)
 65   V(1)=AKK*BLL-1.
      V(2)=A(KK,K)*BLL
      V(3)=AKK*B(L,LL)
      V(4)=A(KK,K)*B(L,LL)
      V(5)=A(K,KK)*BLL
      V(6)=A(KK,KK)*BLL-1.
      V(7)=A(K,KK)*B(L,LL)
      V(8)=A(KK,KK)*B(L,LL)
      V(9)=AKK*B(LL,L)
      V(10)=A(KK,K)*B(LL,L)
      V(11)=AKK*B(LL,LL)-1.
      V(12)=A(KK,K)*B(LL,LL)
      V(13)=A(K,KK)*B(LL,L)
      V(14)=A(KK,KK)*B(LL,L)
      V(15)=A(K,KK)*B(LL,LL)
      V(16)=A(KK,KK)*B(LL,LL)-1.
      W(1)=C(K,L)
      W(2)=C(KK,L)
      W(3)=C(K,LL)
      W(4)=C(KK,LL)
      NDS=NDIM
      NDIM=4
      NDIM1=NDIM+1
      CALL DOOLIT(4,V,W,1,ISG)
      NDIM=NDS
      NDIM1=NDIM+1
 95   K=K+DK
      IF(K.LE.NA) GO TO 10
      L=L+DL
      IF(L.LE.NB) GO TO 5
      RETURN
 99   WRITE(KOUT,101)
      RETURN
 101  FORMAT("0* * * ERROR IN CGT SOLUTION: A11->A23")
C END SUBROUTINE SLVSHR
      END
      SUBROUTINE ENORM(A,NR,NC,ENRM)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      DIMENSION A(1)
      ENRM=0.
      NE=NC*NDIM
      DO 10 I=1,NR
      DO 10 J=I,NE,NDIM
   10 ENRM=ENRM+A(J)*A(J)
      ENRM=SQRT(ENRM)
      RETURN
C END SUBROUTINE ENORM
      END
      SUBROUTINE CGTKX(A11,A13,A21,A23,A12,A22,RELA11,RELA12,RELA13,REL)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/NDIMC/NNC,NRC,NPC
      DIMENSION A11(1),A13(1),A21(1),A23(1),A12(1),A22(1),
     1 RELA11(1),RELA12(1),RELA13(1),REL(1)
      NDIM=NRD
      NDIM1=NDIM+1
      CALL FMMUL(REL,A11,NRD,NND,NNC,RELA11)
      CALL MADD1(NRD,NNC,RELA11,A21,RELA11,1.)
      CALL MATLST(RELA11,NRD,NNC,"KXM",KLIST)
      CALL MATLST(RELA11,NRD,NNC,"KXM",KTERM)
      CALL FMMUL(REL,A12,NRD,NND,NRC,RELA12)
      CALL MADD1(NRD,NRC,RELA12,A22,RELA12,1.)
      CALL MATLST(RELA12,NRD,NRC,"KXU",KLIST)
      CALL MATLST(RELA12,NRD,NRC,"KXU",KTERM)
      IF(NDD.LT.1) RETURN
      CALL FMMUL(REL,A13,NRD,NND,NDD,RELA13)
      CALL MADD1(NRD,NDD,RELA13,A23,RELA13,1.)
      CALL MATLST(RELA13,NRD,NDD,"KXN",KLIST)
      CALL MATLST(RELA13,NRD,NDD,"KXN",KTERM)
      RETURN
C END SUBROUTINE CGTKX
      END
      SUBROUTINE CEVAL
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/INOU/KIN,KOUT,KPUNCH
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/SYSMTX/NVSM,SM(1)
      COMMON/ZMTX1/NVZM,ZM1(1)
      COMMON/ZMTX2/ZM2(1)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/NDIMC/NNC,NRC,NPC
      COMMON/LREGPI/LXDW,LUDW,LPHCL,LGC1,LGC2,LEL,LEE
      COMMON/CREGPI/NVRPI,RPI(1)
      DIMENSION NPLOT(2),NVPLOT(10),NS(6),LSCL(2),ITITLE(5)
      DATA NO/1HN/
      WRITE(KLIST,110)
  110 FORMAT(////11X,5("* "),"CONTROLLER EVALUATION",5(" *")////)
      IPOLE=1
   10 NVOUT=NRD+NPD+1
```

```
      IF(LPLCGT) 17,17,15
   15 WRITE 106
      READ*,IUM,VUM
      IF(IUM.LT.1) GO TO 70
      IF(IUM.GT.NRC) GO TO 15
      NVOUT=NVOUT+NPC
      NP=NNC
      GO TO 18
   17 IF(IPOLE.EQ.1) CALL POLES(RPI(LPHCL),NNPR,4,ZM1,ZM2)
      NP=0
   18 CALL VOUTIC(SM,NVPLOT,NPLOT,NVOUT,LSCL)
      IF(NVOUT.EQ.0) GO TO 70
   20 WRITE 108
      READ*,TEND
      IF(TEND) 20,20,25
   25 LVX0=NVOUT+1
      LX0=LVX0+NVOUT
      LX1=LX0+NPLD
      LXM0=LX1+NPLD
      LXM1=LXM0+NP
      NP=LXM1+NP
      DO 26 I=LVX0,NP
   26 SM(I)=0.
      CALL CTRESP(SM(LVX0),SM,SM(LX0),SM(LX1),SM(LXM0),SM(LXM1),
     1 ZM1,NVOUT,TEND,IUM,VUM,NST)
      WRITE(KTERM,101)
      READ(KIN,102) ITITLE
      M=50*NST
      DO 40 I=1,2
      NS(1)=1
      DO 28 J=2,6
   28 NS(J)=NS(J-1)+51
      NP=NPLOT(I)
      IF(NP.EQ.0) GO TO 40
      NPP1=NP+1
      REWIND KPLOT
      NSV=5*I-4
      CALL RPLOTF(ZM1,NVOUT,IERR)
      CALL STRPLT(SM,ZM1,NS,NVPLOT(NSV),NP,NVOUT)
      DO 35 J=1,M
      CALL RPLOTF(ZM1,NVOUT,IERR)
      IF(IERR.EQ.1) GO TO 40
      IF(MOD(J,NST).NE.0) GO TO 35
      DO 30 K=1,NPP1
   30 NS(K)=NS(K)+1
      CALL STRPLT(SM,ZM1,NS,NVPLOT(NSV),NP,NVOUT)
   35 CONTINUE
      CALL PLOTLP(51,NP,SM,LSCL(I),1,0,KTERM,ITITLE)
   40 CONTINUE
      NVM=NVOUT-1
      M=NVM/5
      NE=5*M
      IF(M.EQ.0) GO TO 56
      DO 55 I=1,NE,5
      NS(1)=1
      DO 42 J=2,6
 42   NS(J)=NS(J-1)+101
      REWIND KPLOT
      NVS=I-1
      DO 45 J=1,5
 45   NVPLOT(J)=NVS+J
      DO 50 J=1,101
      CALL RPLOTF(ZM1,NVOUT,IERR)
      IF(IERR.EQ.1) GO TO 55
      CALL STRPLT(SM,ZM1,NS,NVPLOT,5,NVOUT)
      DO 48 K=1,6
 48   NS(K)=NS(K)+1
 50   CONTINUE
      CALL PLOTLP(101,5,SM,1,1,1,KLIST,ITITLE)
 55   CONTINUE
 56   NVM=NVM-NE
      IF(NVM.LT.1) GO TO 70
      NPP1=NVM+1
      NS(1)=1
      DO 57 I=2,6
 57   NS(I)=NS(I-1)+101
      DO 58 I=1,NVM
 58   NVPLOT(I)=NE+I
      REWIND KPLOT
      DO 65 I=1,101
      CALL RPLOTF(ZM1,NVOUT,IERR)
      IF(IERR.EQ.1) GO TO 70
      CALL STRPLT(SM,ZM1,NS,NVPLOT,NVM,NVOUT)
      DO 60 J=1,NPP1
 60   NS(J)=NS(J)+1
 65   CONTINUE
      CALL PLOTLP(101,NVM,SM,1,1,1,KLIST,ITITLE)
 70   WRITE 104
      READ 111,IANS
      IF(IANS.EQ.NO) RETURN
      IPOLE=0
      GO TO 10
 101  FORMAT(" +",10("-")," ENTER TITLE IN GIVEN FIELD ",10("-"),"+"/)
 102  FORMAT(5A10)
 104  FORMAT("OMORE TIME RESPONSE RUNS (Y OR N) >")
 106  FORMAT("OENTER MODEL INPUT AND STEP VALUE : 1 >")
 108  FORMAT(" ENTER TIME DURATION FOR RESPONSE, IN SECONDS >")
 111  FORMAT(A3)
C END SUBROUTINE CEVAL
      END
      SUBROUTINE VOUTIC(VIC,NVPLOT,NPLOT,NVOUT,LSCL)
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/NDIMC/NNC,NRC,NPC
      COMMON/NDIMT/NNT,NRT,NMT,NWT
      DIMENSION NPLOT(1),NVPLOT(1),VIC(1),IOUT(5),LSCL(2)
      DATA IOUT/1HX,1HY,1HU,1HM,1HD/
      IF(LTEVAL) 2,2,5
 2    NVS=NND
```

```
      NV=NPLD
      GO TO 8
5     NV=NNT
      NVS=NV
8     NVOUT=NVOUT+NV
      DO 9 I=1,NVOUT
9     VIC(I)=0.
      NVU=NV+NPD
      NVM=NVU+NRD
10    WRITE 101,NVS
101   FORMAT("0ENTER STATE AND IC VALUE (0/ TERMINATES): ",I2," >")
12    READ*,IV,V
      IF(IV.LT.1) GO TO 15
      IF(IV.GT.NVS) GO TO 10
      VIC(IV)=V
      GO TO 12
15    IF((LFLCGT.LT.1).OR.(LTEVAL.EQ.1).OR.(NDD.LT.1)) GO TO 25
      LD=1
18    WRITE 102,NDD
102   FORMAT(" ENTER DISTURBANCE IC VALUE (0/ TERMINATES): ",I2," >")
20    READ*,IV,V
      IF(IV.LT.1) GO TO 26
      IF(IV.GT.NDD) GO TO 18
      VIC(NND+IV)=V
      GO TO 20
25    LD=0
26    WRITE 103
103   FORMAT(" 2 PLOTS OF 5 VARIABLES MAY BE PRINTED AT THE TERMINAL --
     1SPECIFY NUMBER FOR EACH (N1,N2) >")
      READ*,NPLOT(1),NPLOT(2)
      IF(NPLOT(1).GT.5) NPLOT(1)=5
      IF(NPLOT(2).GT.5) NPLOT(2)=5
      IF((NPLOT(1).GT.0).OR.(NPLOT(2).GT.0)) GO TO 27
      NVOUT=0
      RETURN
27    WRITE 104
104   FORMAT(" ENTER OUTPUTS BY TYPE AND INDEX IN 2 ENTRIES--TYPES ARE"/
     1 " STATE : 'X'"/" OUTPUT : 'Y'"/" INPUT : 'U'")
      IF(LFLCGT) 30,30,28
28    WRITE 105
105   FORMAT(" MODEL : 'M'")
      IF(LD.EQ.1) WRITE 106
106   FORMAT(" DISTURBANCE : 'D'")
30    DO 40 I=1,2
      NC=NPLOT(I)
      IF(NC.LT.1) GO TO 40
      LSCL(I)=1
      NS=5*(I-1)
      WRITE 107,I
107   FORMAT("0PLOT ",I2)
      DO 39 J=1,NC
      NSP=NS+J
31    WRITE 108,J
108   FORMAT(" OUTPUT ",I2," >")
      READ 111,IV
      WRITE 113
 113  FORMAT(11X,">")
      READ*,IO
      IF(IV.NE.IOUT(1)) GO TO 32
      IF(IO.GT.NVS) GO TO 38
      NVPLOT(NSP)=IO
      GO TO 39
 32   IF(IV.NE.IOUT(2)) GO TO 321
      IF(IO.GT.NPD) GO TO 38
      NVPLOT(NSP)=NV+IO
      GO TO 39
 321  IF(IV.NE.IOUT(3)) GO TO 33
      IF(IO.GT.NRD) GO TO 38
      NVPLOT(NSP)=NVU+IO
      GO TO 39
 33   IF(LFLCGT.LT.1) GO TO 31
      IF(IV.NE.IOUT(4)) GO TO 34
      IF(IO.GT.NPC) GO TO 38
      NVPLOT(NSP)=NVM+IO
      LSCL(I)=-1
      GO TO 39
 34   IF(LD.NE.1) GO TO 31
      IF(IV.NE.IOUT(5)) GO TO 31
      IF(IO.GT.NDD) GO TO 38
      NVPLOT(NSP)=NVS+IO
      GO TO 39
 38   WRITE 109
 109  FORMAT(" INDEX TOO LARGE")
      GO TO 31
 39   CONTINUE
 40   CONTINUE
      NVM1=NVOUT-1
      NP=0
      DO 50 I=1,NVM1
      M=MOD((I-1),5)+1
      IF(M.GT.1) GO TO 41
      NP=NP+1
      WRITE(KLIST,110) NP
 110  FORMAT("OPLOT ",I2)
 41   IF(I.GT.NVS) GO TO 42
      IV=IOUT(1)
      IO=I
      GO TO 45
 42   IF(I.GT.NV) GO TO 43
      IV=IOUT(5)
      IO=I-NVS
      GO TO 45
 43   IF(I.GT.NVU) GO TO 441
      IV=IOUT(2)
      IO=I-NV
      GO TO 45
 441  IF(I.GT.NVM) GO TO 44
      IV=IOUT(3)
      IO=I-NVU
      GO TO 45
 44   IV=IOUT(4)
      IO=I-NVM
 45   WRITE(KLIST,112) M,IV,IO
 50   CONTINUE
      RETURN
 111  FORMAT(A1)
 112  FORMAT("    OUTPUT ",I2,": ",A1,I2)
C END SUBROUTINE VOUTIC
      END
      SUBROUTINE CTRESP(VX0,VX1,X0,X1,XM0,XM1,ZM1,NVOUT,TEND,IUM,VUM,
     1 NST)
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/LOCD/LAP,LGP,LPHI,LBD,LEX,LPHD,LQ,LQN,LQD,LC,LDY,LEY,LHP,LR
      COMMON/DSNMTX/NVDM,NODY,NOEY,DM(1)
      COMMON/NDIMC/NNC,NRC,NPC
      COMMON/LOCC/LPHC,LBDC,LCC,LDC
      COMMON/CMDMTX/NVCM,NEWCM,NODC,CM(1)
      COMMON/LOCT/LPHT,LBDT,LQDT,LHT,LRT,LTDT,LTNT
      COMMON/TRUMTX/NVTM,TM(1)
      COMMON/LREGPI/LXDW,LUDW,LPHCL,LGC1,LGC2,LEL,LEE
      COMMON/CREGPI/NVRPI,RPI(1)
      DIMENSION VX0(1),VX1(1),X0(1),X1(1),XM0(1),XM1(1),ZM1(1)
      NSTPO=.01*TEND/TSAMP+.5
      NST=2
      IF(NSTPO.GE.1) GO TO 1
      NSTPO=1
      NST=1
 1    NSTEPS=100*NSTPO
      IF(LFLCGT.EQ.0) GO TO 2
      LMO=NVOUT-NPC
      IF(NDD.EQ.0) GO TO 4
      LDCGT=1
      GO TO 5
 2    LMO=NVOUT
 4    LDCGT=0
 5    LU=LMO-NRD
      LSO=LU-NPD
      NVX=LSO-1
      IF(LTEVAL)6,6,10
 6    DO 7 I=1,NVX
 7    X1(I)=VX1(I)
      GO TO 12
 10   CALL XFDT(VX1,X1,LDCGT)
 12   NNDP1=NND+1
      REWIND KPLOT
      CALL YDSN(X1,VX1(LU),DM(LC),DM(LDY),LDCGT,VX1(LSO))
      IF(LFLCGT.EQ.1) CALL YCMD(XM1,IUM,VUM,CM(LCC),CM(LDC),
     1 VX1(LMO))
      CALL WPLOTF(VX1,NVOUT)
      DO 100 IT=1,NSTEPS
      CALL URPI(RPI(LGC1),RPI(LGC2),DM(LC),DM(LDY),X0,X1,VX0(LU),
     1 VX1(LU))
      IF(LFLCGT) 20,20,15
 15   CALL UCGT(VXO(LU),VX1(LU),XMO,XM1,XO(NNDP1),ZM1,IUM,VUM,IT)
      CALL CUPDAT(XMO,XM1,IUM,VUM)
 20   CALL FTMTX(VX1(LU),VXO(LU),NRD,1)
      CALL FTMTX(X1,XO,NPLD,1)
      IF(LTEVAL) 25,25,30
 25   CALL DUPDAT(DM(LPHI),DM(LBD),DM(LPHD),DM(LEX),XO,X1,
     1 VX1,VXO(LU),LDCGT,NNDP1)
      GO TO 35
 30   CALL TUPDAT(TM(LPHT),TM(LBDT),VXO,VX1,VXO(LU))
      CALL XFDT(VX1,X1,LDCGT)
 35   IF(MOD(IT,NSTPO).NE.0) GO TO 100
      VX1(NVOUT)=TSAMP*FLOAT(IT)
      CALL YDSN(X1,VX1(LU),DM(LC),DM(LDY),LDCGT,VX1(LSO))
      IF(LFLCGT.EQ.1) CALL YCMD(XM1,IUM,VUM,CM(LCC),CM(LDC),
     1 VX1(LMO))
      CALL WPLOTF(VX1,NVOUT)
 100  CONTINUE
      ENDFILE KPLOT
      RETURN
C END SUBROUTINE CTRESP
      END
      SUBROUTINE DUPDAT(PHI,BD,PHID,EX,XO,X1,VX1,UO,LDCGT,NNDP1)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      DIMENSION PHI(1),BD(1),PHID(1),EX(1),XO(1),X1(1),VX1(1),UO(1)
      NDIM=NND
      NDIM1=NDIM+1
      CALL FMMUL(BD,UO,NND,NRD,1,X1)
      CALL MMULS(PHI,XO,NND,NND,1,X1)
      IF(LDCGT.EQ.0) GO TO 10
      CALL FMMUL(PHID,XO(NNDP1),NDD,NDD,1,X1(NNDP1))
      CALL MMULS(EX,X1(NNDP1),NND,NDD,1,X1)
 10   CALL FTMTX(X1,VX1,NPLD,1)
      RETURN
C END SUBROUTINE DUPDAT
      END
      SUBROUTINE CUPDAT(XMO,XM1,IUM,VUM)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/NDIMC/NNC,NRC,NPC
      COMMON/LOCC/LPHC,LBDC,LCC,LDC
      COMMON/CMDMTX/NVCM,NEWCM,NODC,CM(1)
      DIMENSION XMO(1),XM1(1)
      NDIM=NNC
      NDIM1=NDIM+1
      CALL FTMTX(XM1,XMO,NNC,1)
      L1=LADDR(NNC,1,IUM)+LBDC-1
      CALL VSCALE(XM1,CM(L1),NNC,VUM)
      CALL MMULS(CM(LPHC),XMO,NNC,NNC,1,XM1)
      RETURN
C END SUBROUTINE CUPDAT
      END
      SUBROUTINE TUPDAT(PHI,BD,VXO,VX1,UO)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/NDIMT/NNT,NRT,NMT,NWT
      DIMENSION PHI(1),BD(1),VXO(1),VX1(1),UO(1)
      NDIM=NNT
      NDIM1=NDIM+1
      CALL FTMTX(VX1,VX0,NNT,1)
      CALL FMMUL(BD,U0,NNT,NRT,1,VX1)
      CALL MMULS(PHI,VX0,NNT,NNT,1,VX1)
      RETURN
C END SUBROUTINE TUPDAT
      END
      SUBROUTINE XFDT(V,X,LDCGT)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/NDIMT/NNT,NRT,NMT,NWT
      COMMON/LOCT/LPHT,LBDT,LQDT,LHT,LRT,LTDT,LTNT
      COMMON/TRUMTX/NVTM,TM(1)
      DIMENSION V(1),X(1)
      CALL FMMUL(TM(LTDT),V,NND,NNT,1,X)
      IF(LDCGT.EQ.0) RETURN
      CALL FMMUL(TM(LTNT),V,NDD,NNT,1,X(NND+1))
      RETURN
C END SUBROUTINE XFDT
      END
      SUBROUTINE URPI(RGC1,RGC2,C,DY,X0,X1,U0,U1)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      DIMENSION RGC1(1),RGC2(1),C(1),DY(1),X0(1),X1(1),U0(1),U1(1)
      CALL YDSN(X0,U0,C,DY,0,U1)
 10   CALL VSCALE(U1,U1,NRD,-1.)
      CALL MMULS(RGC2,U1,NRD,NRD,1,U0)
      DO 12 I=1,NPLD
 12   X0(I)=X0(I)-X1(I)
      CALL FMMUL(RGC1,X0,NRD,NND,1,U1)
      CALL VADD(NRD,1.,U1,U0)
      RETURN
C END SUBROUTINE URPI
      END

      SUBROUTINE UCGT(U0,U1,XM0,XM1,DDIF,ZM1,IUM,VUM,IT)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/NDIMC/NNC,NRC,NPC
      COMMON/LOCC/LPHC,LBDC,LCC,LDC
      COMMON/CMDMTX/NVCM,NEWCM,NODC,CM(1)
      COMMON/LREGPI/LXDW,LUDW,LPHCL,LGC1,LGC2,LEL,LEE
      COMMON/CREGPI/NVRPI,RPI(1)
      COMMON/LCGT/LA11,LA13,LA21,LA23,LA12,LA22,LELA11,LELA12,LELA13
      COMMON/CCGT/NVCGT,CGT(1)
      DIMENSION U0(1),U1(1),XM0(1),XM1(1),DDIF(1),ZM1(1)
      CALL YCMD(XM0,IUM,VUM,CM(LCC),CM(LDC),U0)
      IF(IT.GT.1) GO TO 10
      I=LELA12+LADDR(NPD,1,IUM)-1
      CALL MADD1(NPD,1,U1,CGT(I),U1,VUM)
 10   CALL MMULS(RPI(LGC2),U0,NDIM,NDIM,1,U1)
      DO 12 I=1,NNC
 12   XM0(I)=XM1(I)-XM0(I)
      CALL FMMUL(CGT(LELA11),XM0,NPD,NNC,1,U0)
      CALL VADD(NDIM,1.,U1,U0)
      IF(NDD.EQ.0) RETURN
      DO 14 I=1,NDD
14    DDIF(I)=-DDIF(I)
      CALL MMULS(CGT(LELA13),DDIF,NPD,NDD,1,U1)
      RETURN
C END SUBROUTINE UCGT
      END
      SUBROUTINE YDSN(X,U,C,D,LDCGT,Y)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/LOCD/LAP,LGP,LPHI,LBD,LEX,LPHD,LQ,LQN,LQD,LC,LDY,LEY,LHP,LR
      COMMON/DSNMTX/NVDM,NODY,NOEY,DM(1)
      DIMENSION X(1),U(1),C(1),D(1),Y(1)
      NDIM=NPD
      NDIM1=NDIM+1
      CALL FMMUL(C,X,NPD,NND,1,Y)
      IF(NODY.EQ.1) GO TO 10
      CALL MMULS(D,U,NPD,NRD,1,Y)
10    IF((LDCGT.EQ.0).OR.(NOEY.EQ.1)) RETURN
      CALL MMULS(DM(LEY),X(NND+1),NPD,NDD,1,Y)
      RETURN
C END SUBROUTINE YDSN
      END
      SUBROUTINE YCMD(X,IU,VU,C,D,Y)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/NDIMC/NNC,NRC,NPC
      COMMON/CMDMTX/NVCM,NEWCM,NODC,CM(1)
      DIMENSION X(1),C(1),D(1),Y(1)
      NDIM=NPC
      NDIM1=NDIM+1
      CALL FMMUL(C,X,NPC,NNC,1,Y)
      IF(NODC.EQ.1) RETURN
      L1=LADDR(NPC,1,IU)
      CALL MADD1(NPC,1,Y,D(L1),Y,VU)
      RETURN
C END SUBROUTINE YCMD
      END
      SUBROUTINE FLTRK(IFLTR)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/MAIN2/COM2(1)
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/SYSMTX/NVSM,SM(1)
      COMMON/ZMTX1/NVZM,ZM1(1)
      COMMON/ZMTX2/ZM2(1)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/LOCD/LAP,LGP,LPHI,LBD,LEX,LPHD,LQ,LQN,LQD,LC,LDY,LEY,LHP,LR
      COMMON/DSNMTX/NVDM,NODY,NOEY,DM(1)
      COMMON/LKF/LEADSN,LFLTRK,LFCOV
      COMMON/CKF/NVFLT,FLT(1)
      IF(NWPNWD.GT.0) GO TO 1
      WRITE(KTERM,108)
108   FORMAT("0NO DRIVING NOISES - - FILTER DESIGN ABORT")
      RETURN
1     IF(NMD.GT.0) GO TO 2
```

```
      WRITE(KTERM,109)
  109 FORMAT("ONO MEASUREMENTS - - FILTER DESIGN ABORT")
      RETURN
    2 WRITE(KLIST,110)
  110 FORMAT(////11X,5("* "),"FILTER DESIGN",5(" *")////)
      NSIZE=NPLD*(1+NPLD+NMD)
      IF(NSIZE.LE.NVFLT) GO TO 3
      WRITE 101,NSIZE
  101 FORMAT("0INSUFFICIENT MEMORY /CKF/, NEED: ",I4)
      LABORT=NSIZE
      RETURN
    3 NDIM=NPLD
      NDIM1=NDIM+1
    5 IF(NWD.EQ.0) GO TO 12
      IF(IFLTR.LE.0) GO TO 6
      WRITE 105,NWD
  105 FORMAT(" ENTER STATE NOISE STRENGTHS: ",I2)
      CALL RQWGTS(DM(LQ),NWD,0)
    6 CALL DVCTOR(NWD,DM(LQ),ZM1)
      CALL MATLST(ZM1,NWD,1,"Q",KTERM)
   10 CALL MATLST(DM(LQ),NWD,NWD,"Q",KLIST)
   12 IF(NWDD.EQ.0) GO TO 18
      IF(IFLTR.LE.0) GO TO 13
      WRITE 106,NWDD
  106 FORMAT(" ENTER DISTURBANCE NOISE STRENGTHS: ",I2)
      CALL RQWGTS(DM(LQN),NWDD,0)
   13 CALL DVCTOR(NWDD,DM(LQN),ZM1)
      CALL MATLST(ZM1,NWDD,1,"QN",KTERM)
   15 CALL MATLST(DM(LQN),NWDD,NWDD,"QN",KLIST)
   18 CALL QDSCRT(DM(LQ),DM(LQN),ZM1,ZM2)
      IF(IFLTR.LE.0) GO TO 19
      WRITE 107,NMD
  107 FORMAT(" ENTER MEASUREMENT NOISE STRENGTHS: ",I2)
      CALL RQWGTS(DM(LR),NMD,0)
   19 CALL DVCTOR(NMD,DM(LR),ZM1)
      CALL MATLST(ZM1,NMD,1,"R",KTERM)
   20 CALL MATLST(DM(LR),NMD,NMD,"R",KLIST)
   25 CALL TFRMTX(DM(LHP),SM,NMD,NDIM,2)
      CALL TRANS2(NMD,NDIM,SM,ZM1)
      LFCOV=LFLTRK+NDIM*NMD
      CALL DVCTOR(NMD,DM(LR),FLT(LFCOV))
      CALL KFLTR(NDIM,NMD,FLT,ZM1,DM(LQD),FLT(LFCOV),ZM2,
     1 FLT(LFLTRK),SM)
      CALL TFRMTX(SM,COM2,NDIM,NDIM,2)
      IA=1
      DO 30 I=1,NPLD
      FLT(LFCOV-1+I)=SQRT(ZM2(IA))
   30 IA=IA+NDIM1
      CALL MATLST(FLT(LFLTRK),NDIM,NMD,"KF",KLIST)
      CALL MATLST(FLT(LFLTRK),NDIM,NMD,"KF",KTERM)
      IFLTR=1
      LFLKF=1
  111 FORMAT(A3)
      RETURN
C END SUBROUTINE FLTRK
```

```
      END
      SUBROUTINE FEVAL
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/MAIN2/COM2(1)
      COMMON/INOU/KIN,KOUT,KPUNCH
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/SYSMTX/NVSM,SM(1)
      COMMON/ZMTX1/NVZM,ZM1(1)
      COMMON/ZMTX2/ZM2(1)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/LOCD/LAP,LGP,LPHI,LBD,LEX,LPHD,LQ,LQN,LQD,LC,LDY,LEY,LHP,LR
      COMMON/DSNMTX/NVDM,NODY,NOEY,DM(1)
      COMMON/NDIMT/NNT,NRT,NMT,NWT
      COMMON/LOCT/LPHT,LBDT,LQDT,LHT,LRT,LTDT,LTNT
      COMMON/TRUMTX/NVTM,TM(1)
      COMMON/LKF/LEADSN,LFLTRK,LFCOV
      COMMON/CKF/NVFLT,FLT(1)
      DIMENSION ITITLE(5),NS(3),NVPLOT(2)
      IF(NWT.GT.0) GO TO 1
      WRITE(KTERM,108)
  108 FORMAT("ONO TRUTH MODEL DRIVING NOISE - - FILTER EVALUATION ABORTE
     1D")
      RETURN
    1 WRITE(KLIST,110)
  110 FORMAT(////11X,5("* "),"FILTER EVALUATION",5(" *")////)
      CALL FMMUL(COM2,FLT(LEADSN),NPLD,NPLD,NPLD,SM)
      CALL POLES(SM,NPLD,5,ZM1,ZM2)
      NA=NNT+NPLD
      NSIZE=NA*NA
      IF(NSIZE.LE.NVSM) GO TO 8
      WRITE 101,NSIZE
  101 FORMAT("OINSUFFICIENT MEMORY /SYSMTX/, NEED: ",I4)
      GO TO 9
    8 IF(NSIZE.LE.NVZM) GO TO 10
      WRITE 103,NSIZE
  103 FORMAT("OINSUFFICIENT MEMORY /ZMTX1/,/ZMTX2/, NEED: ",I4)
    9 LABORT=NSIZE
      RETURN
   10 CALL ZPART(SM,1,NSIZE,1)
      NDIM=NPLD
      NDIM1=NDIM+1
      CALL TFRMTX(TM(LRT),ZM1,NMD,NMD,2)
      CALL MAT3(NPLD,NMD,FLT(LFLTRK),ZM1,COM1)
      NVOUT=2*NPLD+1
      REWIND KPLOT
      CALL DACOV(SM,FLT(LFCOV),ZM1,ZM2,NA,NVOUT,0.)
      DO 20 IT=1,50
      TIME=TSAMP*FLOAT(IT)
      CALL ACOVUD(SM,TM(LQDT),COM1,TM(LPHT),FLT(LEADSN),
     1 COM2,ZM1,ZM2)
      CALL DACOV(SM,FLT(LFCOV),ZM1,ZM2,NA,NVOUT,TIME)
   20 CONTINUE
      ENDFILE KPLOT
      WRITE(KTERM,104)
      READ(KIN,102) ITITLE
      DO 50 I=1,NPLD
      REWIND KPLOT
      NS(1)=1
      NS(2)=52
      NS(3)=103
      NVPLOT(1)=I+I-1
      NVPLOT(2)=I+I
      DO 40 J=1,51
      CALL RPLOTF(ZM1,NVOUT,IERR)
      IF(IERR.EQ.1) GO TO 50
      CALL STRPLT(SM,ZM1,NS,NVPLOT,2,NVOUT)
      DO 35 K=1,3
35    NS(K)=NS(K)+1
40    CONTINUE
      WRITE 107,ZM1(NVPLOT(1)),I,ZM1(NVPLOT(2))
107   FORMAT("0FINAL RMS ERRORS : TRUE = ",1PE15.7/"  (STATE",I3,
     1 ")",4X,"COMPUTED = ",1PE15.7)
      CALL PLOTLP(51,2,SM,-1,1,0,KLIST,ITITLE)
      WRITE(KLIST,106) I
106   FORMAT("0   STATE : ",I2//4X,"SYMBOL 1 : TRUE ERROR"/
     1 4X,"SYMBOL 2 : COMPUTED ERROR "/)
50    CONTINUE
      RETURN
104   FORMAT(" +",10("-")," ENTER TITLE IN GIVEN FIELD ",10("-"),"+"/)
102   FORMAT(5A10)
C END SUBROUTINE FEVAL
      END
      SUBROUTINE DACOV(PCA,PC,ZM1,ZM2,NA,NVOUT,TIME)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/NDIMT/NNT,NRT,NMT,NWT
      COMMON/LOCT/LPHT,LBDT,LQDT,LHT,LRT,LTDT,LTNT
      COMMON/TRUMTX/NVTM,TM(1)
      DIMENSION PCA(1),PC(1),ZM1(1),ZM2(1)
      NDIM=NA
      NDIM1=NDIM+1
      CALL TFRMTX(TM(LTDT),ZM1,NND,NNT,2)
      IF(NDD.LT.1) GO TO 5
      IA=LADDR(NA,NND+1,1)
      CALL TFRMTX(TM(LTNT),ZM1(IA),NDD,NNT,2)
5     CALL MSCALE(ZM1,ZM1,NPLD,NNT,-1.)
      IA=LADDR(NA,1,NNT+1)
      CALL IDNT(NPLD,ZM1(IA),1.)
      CALL MAT3(NPLD,NA,ZM1,PCA,ZM2)
      WRITE(KLIST,101) TIME
101   FORMAT("0'TRUE' DESIGN ERROR COVARIANCE AT TIME = ",F6.4)
      CALL MATIO(ZM2,NPLD,NPLD,3)
      IA=1
      DO 10 I=1,NPLD
      NS=I+I
      ZM1(NS-1)=SQRT(ZM2(IA))
      ZM1(NS)=PC(I)
10    IA=IA+NDIM1
```

```
      ZM1(NVOUT)=TIME
      CALL WPLOTF(ZM1,NVOUT)
      RETURN
C END SUBROUTINE DACOV
      END
      SUBROUTINE ACOVUD(PC,QD,RKRKT,PHIT,PHI,RIMKH,ZM1,ZM2)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/NDIMT/NNT,NRT,NMT,NWT
      COMMON/LOCT/LPHT,LBDT,LQDT,LHT,LRT,LTDT,LTNT
      COMMON/TRUMTX/NVTM,TM(1)
      COMMON/LKF/LEADSN,LFLTRK,LFCOV
      COMMON/CKF/NVFLT,FLT(1)
      DIMENSION PC(1),QD(1),RKRKT(1),PHIT(1),PHI(1),RIMKH(1),
     1 ZM1(1),ZM2(1)
      L1=LADDR(NDIM,1,NNT+1)
      CALL ZPART(ZM2(L1),NNT,NPLD,NDIM)
      CALL TFRMTX(PHIT,ZM2,NNT,NNT,2)
      L1=LADDR(NDIM,NNT+1,NNT+1)
      CALL TFRMTX(PHI,ZM2(L1),NPLD,NPLD,2)
      L2=LADDR(NDIM,NNT+1,1)
      CALL ZPART(ZM2(L2),NPLD,NNT,NDIM)
      CALL MAT3(NDIM,NDIM,ZM2,PC,ZM1)
      CALL FPADD(ZM1,NDIM,QD,NNT,NNT,1,PC)
      CALL IDNT(NNT,ZM2,1.)
      CALL FMMUL(FLT(LFLTRK),TM(LHT),NPLD,NMD,NNT,ZM1)
      CALL TFRMTX(ZM1,ZM2(L2),NPLD,NNT,2)
      CALL TFRMTX(RIMKH,ZM2(L1),NPLD,NPLD,2)
      CALL MAT3(NDIM,NDIM,ZM2,PC,ZM1)
      CALL FPADD(ZM1,NDIM,RKRKT,NPLD,NPLD,L1,PC)
      RETURN
C END SUBROUTINE ACOVUD
      END
      SUBROUTINE FPADD(X,NX,Y,NRY,NCY,LADDR,Z)
      DIMENSION X(1),Z(1),Y(NRY,NCY)
      CALL FTMTX(X,Z,NX,NX)
      LAM1=LADDR-1
      DO 10 I=1,NCY
      LA1=LAM1+NX*(I-1)
      DO 10 J=1,NRY
      LA1=LA1+1
 10   Z(LA1)=Z(LA1)+Y(J,I)
      RETURN
C END SUBROUTINE FPADD
      END
      SUBROUTINE RSYS(A,L,ND,ITYPE,IWRT)
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/SYSMTX/NVSM,SM(1)
      COMMON/AMC/AM(1)
      COMMON/BDG/BD(1)
      DIMENSION A(1),L(1),ND(1),NAD(14,2),IND(7,3),NTYP(2,3),NTITLE(3),
     1 NMAT(14,3)
      DATA NTYP/7,14,3,4,4,8/
      DATA NO/1HN/
      DATA IND/1HN,1HR,1HP,1HM,1HD,1HW,2HWD,2HNM,2HRM,2HPM,4(1H ),
     1 2HNT,2HRT,2HMT,2HWT,1H ,1H ,1H /
      DATA NTITLE/6HDESIGN,7HCOMMAND,5HTRUTH/
      DATA NMAT/1HA,1HB,2HEX,1HG,1HQ,1HC,2HDY,2HEY,1HH,2HHN,1HR,2HAN,
     1 2HGN,2HQN,2HAM,2HBM,2HCM,2HDM,10(1H ),2HAT,2HBT,2HGT,2HQT,2HHT,
     2 2HRT,3HTDT,3HTNT,6(1H )/
      NDM=NTYP(1,ITYPE)
      NAR=NTYP(2,ITYPE)
      NT=NTITLE(ITYPE)
      WRITE(KLIST,110) NT
  110 FORMAT(////11X,5("* "),A7," MODEL",5(" *")////)
    5 WRITE 101,NT
  101 FORMAT("0READ ",A7," MODEL FROM 'DATA' FILE (Y OR N) >")
      READ 111,IANS
      IF(IANS.EQ.NO) GO TO 10
      IF=1
      CALL READFS(A,ND,ITYPE,IERR)
      IF(IERR.EQ.0) GO TO 201
   10 WRITE 102,NT
  102 FORMAT(" ENTER ",A7," MODEL FROM TERMINAL (Y OR N) >")
      READ 111,IANS
      IF(IANS.EQ.NO) GO TO 15
      IF=2
      DO 12 I=1,NDM
      WRITE 112,IND(I,ITYPE)
  112 FORMAT(" ENTER ",A2," >")
   12 READ*,ND(I)
      GO TO 201
   15 IF=3
      IF(ITYPE-2) 16,17,18
   16 CALL DSND(ND)
      GO TO 20
   17 CALL CMDD(ND)
      GO TO 20
   18 CALL TRTHD(ND)
   20 IF(ND(1).GT.0) GO TO 201
      WRITE 114,NT
  114 FORMAT("0",A7," MODEL SUBROUTINE NON-EXISTENT")
      GO TO 5
  201 IF(ITYPE-2) 21,22,23
   21 CALL DSNDM(ND,NAD)
      GO TO 25
   22 CALL CMDDM(ND,NAD)
      GO TO 25
   23 CALL TRTHDM(ND,NAD)
   25 IF(LABORT.EQ.0) GO TO 26
      WRITE 103,NT,LABORT
  103 FORMAT("0INSUFFICIENT MEMORY FOR ",A7," MODEL, NEED: ",I4)
      RETURN
   26 L(1)=1
      DO 30 I=2,NAR
   30 L(I)=L(I-1)+NAD(I-1,1)*NAD(I-1,2)
      NPNTS=L(NAR)+NAD(NAR,1)*NAD(NAR,2)-1
      IF(NPNTS.LE.NVSM) GO TO 34
      WRITE 104,NPNTS
  104 FORMAT("OINSUFFICIENT MEMORY /SYSMTX/, NEED: ",I4)
      LABORT=NPNTS
      RETURN
  34  IF(IF-2) 75,35,50
  35  IZ=1
      DO 40 I=1,NAR
      N1=NAD(I,1)
      N2=NAD(I,2)
      IF((N1.EQ.0).OR.(N2.EQ.0)) GO TO 40
      WRITE 113,NMAT(I,ITYPE)
  113 FORMAT("OENTER ",A3)
      CALL ZMATIN(A(L(I)),N1,N2,IZ)
  40  CONTINUE
      GO TO 75
  50  CALL ZPART(A,1,NPNTS,1)
      IF(ITYPE-2) 55,60,65
  55  CALL DSNM(A(L(1)),A(L(2)),A(L(3)),A(L(4)),A(L(5)),A(L(6)),A(L(7)),
     1 A(L(8)),A(L(9)),A(L(10)),A(L(11)),A(L(12)),A(L(13)),A(L(14)))
      GO TO 75
  60  CALL CMDM(A(L(1)),A(L(2)),A(L(3)),A(L(4)))
      GO TO 75
  65  CALL TRTHM(A(L(1)),A(L(2)),A(L(3)),A(L(4)),A(L(5)),A(L(6)),
     1 A(L(7)),A(L(8)))
  75  IZ=0
  77  WRITE 105
  105 FORMAT("OMODIFY MATRIX ELEMENTS (Y OR N) >")
      READ 111,IANS
      IF(IANS.EQ.NO) GO TO 88
      WRITE 106,(NMAT(I,ITYPE),I=1,NAR)
  106 FORMAT(1X,14(2X,A3))
  78  WRITE 107
  107 FORMAT(" ENTER MATRIX NAME >")
      READ 111,IANS
      DO 80 I=1,NAR
      IF(IANS.EQ.NMAT(I,ITYPE)) GO TO 81
  80  CONTINUE
      GO TO 78
  81  WRITE 116
  116 FORMAT(" LIST MATRIX TO TERMINAL (Y OR N) >")
      READ 111,IANS
      IF(IANS.EQ.NO) GO TO 83
      CALL MATLST(A(L(I)),NAD(I,1),NAD(I,2),NMAT(I,ITYPE),KTERM)
  83  CALL ZMATIN(A(L(I)),NAD(I,1),NAD(I,2),IZ)
      GO TO 77
  88  IF(ITYPE.EQ.2)CALL FTMTX(A(L(1)),AM,ND(1),ND(1))
      IF(ITYPE.EQ.1)CALL FTMTX(A(L(2)),BD,ND(1),ND(2))
      IF(IWRT) 95,92,93
  92  IWRT=1
  93  WRITE 115,NT
  115 FORMAT("OWRITE ",A7," MODEL TO 'SAVE' FILE (Y OR N) >")
      READ 111,IANS
      IF(IANS.EQ.NO) GO TO 95
      CALL WFILED(ITYPE,NPNTS,ND,A)
      IWRT=-1
      WRITE 109,NT
```

```
 109  FORMAT(6X,A7," MODEL WRITTEN TO 'SAVE' FILE")
 95   DO 100 I=1,NAR
      N1=NAD(I,1)
      N2=NAD(I,2)
      IF((N1.EQ.0).OR.(N2.EQ.0)) GO TO 100
      CALL MATLST(A(L(I)),N1,N2,NMAT(I,ITYPE),KLIST)
 100  CONTINUE
 111  FORMAT(A3)
      RETURN
C END SUBROUTINE RSYS
      END
      SUBROUTINE DSND(ND)
      DIMENSION ND(1)
      ND(1)=0
      RETURN
C END SUBROUTINE DSND
      END
      SUBROUTINE CMDD(ND)
      DIMENSION ND(1)
      ND(1)=0
      RETURN
C END SUBROUTINE CMDD
      END
      SUBROUTINE TRTHD(ND)
      DIMENSION ND(1)
      ND(1)=0
      RETURN
C END SUBROUTINE TRTHD
      END
      SUBROUTINE DSNM(A,B,EX,G,Q,C,DY,EY,H,HD,R,AD,GD,QD)
      RETURN
C END SUBROUTINE DSNM
      END
      SUBROUTINE CMDM(AM,BM,CM,DM)
      RETURN
C END SUBROUTINE CMDM
      END
      SUBROUTINE TRTHM(AT,BT,GT,QT,HT,RT,TDT,TNT)
      RETURN
C END SUBROUTINE TRTHM
      END
      SUBROUTINE DSNDM(ND,NAD)
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/DSNMTX/NVDM,NODY,NOEY,DM(1)
      DIMENSION ND(1),NAD(14,2)
      NND=ND(1)
      NRD=ND(2)
      NPD=ND(3)
      NMD=ND(4)
      NDD=ND(5)
      NWD=ND(6)
      NWDD=ND(7)
      NPLD=NND+NDD
      NWPNWD=NWD+NWDD
      NNPR=NND+NRD
      NAD(1,1)=NND
      NAD(2,1)=NND
      NAD(3,1)=NND
      NAD(4,1)=NND
      NAD(5,1)=NWD
      NAD(6,1)=NPD
      NAD(7,1)=NPD
      NAD(8,1)=NPD
      NAD(9,1)=NMD
      NAD(10,1)=NMD
      NAD(11,1)=NMD
      NAD(12,1)=NDD
      NAD(13,1)=NDD
      NAD(14,1)=NWDD
      NAD(1,2)=NND
      NAD(2,2)=NRD
      NAD(3,2)=NDD
      NAD(4,2)=NWD
      NAD(5,2)=NWD
      NAD(6,2)=NND
      NAD(7,2)=NRD
      NAD(8,2)=NDD
      NAD(9,2)=NND
      NAD(10,2)=NDD
      NAD(11,2)=NMD
      NAD(12,2)=NDD
      NAD(13,2)=NWDD
      NAD(14,2)=NWDD
      NSIZE=NPLD*(2*NPLD+NND+NMD+NPD+NWPNWD)+NRD*(NND+NPD)+
     1 NDD*NDD+NMD*NMD+NWD*NWD+NWDD*NWDD
      IF(NSIZE.GT.NVDM) LABORT=NSIZE
      RETURN
C END SUBROUTINE DSNDM
      END
      SUBROUTINE CMDDM(ND,NAD)
      COMMON/NDIMC/NNC,NRC,NPC
      COMMON/CMDMTX/NVCM,NEWCM,NODC,CM(1)
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      DIMENSION ND(1),NAD(14,2)
      NNC=ND(1)
      NRC=ND(2)
      NPC=ND(3)
      NAD(1,1)=NNC
      NAD(2,1)=NNC
      NAD(3,1)=NPC
      NAD(4,1)=NPC
      NAD(1,2)=NNC
      NAD(2,2)=NRC
      NAD(3,2)=NNC
      NAD(4,2)=NRC
      NSIZE=NNC*(NNC+NRC+NPC)+NPC*NRC
      IF(NSIZE.GT.NVCM) LABORT=NSIZE
      RETURN
C END SUBROUTINE CMDDM
```

```
      END
      SUBROUTINE TRTHDM(ND,NAD)
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/NDIMT/NNT,NRT,NMT,NWT
      COMMON/TRUMTX/NVTM,TM(1)
      DIMENSION ND(1),NAD(14,2)
      NNT=ND(1)
      NRT=ND(2)
      NMT=ND(3)
      NWT=ND(4)
      NAD(1,1)=NNT
      NAD(2,1)=NNT
      NAD(3,1)=NNT
      NAD(4,1)=NWT
      NAD(5,1)=NMT
      NAD(6,1)=NMT
      NAD(7,1)=NND
      NAD(8,1)=NDD
      NAD(1,2)=NNT
      NAD(2,2)=NRT
      NAD(3,2)=NWT
      NAD(4,2)=NWT
      NAD(5,2)=NNT
      NAD(6,2)=NMT
      NAD(7,2)=NNT
      NAD(8,2)=NNT
      NSIZE=NNT*(2*NNT+NMD+NRD+NPLD)+NMD*NMD
      IF(NSIZE.GT.NVTM) LABORT=NSIZE
      RETURN
C END SUBROUTINE TRTHDM
      END
      SUBROUTINE ZMATIN(A,NR,NC,IZ)
      DIMENSION A(NR,NC)
      IF(IZ) 10,10,1
 1    DO 5 I=1,NR
      DO 5 J=1,NC
 5    A(I,J)=0.
 10   WRITE 101,NR,NC
 15   READ*,I,J,V
      IF(I.EQ.0) RETURN
      IF((I.LE.NR).AND.(J.LE.NC)) GO TO 20
      WRITE 102
      GO TO 10
 20   A(I,J)=V
      IF(IZ.LT.0) A(J,I)=V
      GO TO 15
 101  FORMAT(" ENTER I,J AND M(I,J)--(0/ WHEN COMPLETE) : "I2," BY "I2,
     1 " >")
 102  FORMAT(" ERROR IN ARRAY INDEX")
C END SUBROUTINE ZMATIN
      END
      SUBROUTINE WFILED(NT,NP,ND,A)
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      DIMENSION ND(15),A(NP)
      DATA IEOI/-1/
      BACKSPACE KSAVE
      WRITE(KSAVE,101) NT,NP
      WRITE(KSAVE,101) (ND(I),I=1,15)
      WRITE(KSAVE,102) (A(I),I=1,NP)
      WRITE(KSAVE,101) IEOI,NP
      RETURN
 101  FORMAT(2I4)
 102  FORMAT(E20.10)
C END SUBROUTINE WFILED
      END
      SUBROUTINE READFS(A,ND,NT,IERR)
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      DIMENSION A(1),ND(15)
      DATA IEOI/-1/
      REWIND KDATA
 5    READ(KDATA,102) IT,NP
      IF(IT.NE.IEOI) GO TO 10
      WRITE 101
      IERR=1
      RETURN
 10   CALL FARRAY(A,ND,NP)
      IF(IT.NE.NT) GO TO 5
      IERR=0
 101  FORMAT("0DATA NOT IN 'DATA' FILE . . .")
 102  FORMAT(2I4)
      RETURN
C END SUBROUTINE READFS
      END
      SUBROUTINE FARRAY(A,ND,NP)
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      DIMENSION A(NP),ND(15)
      READ(KDATA,101) (ND(I),I=1,15)
      READ(KDATA,102) (A(I),I=1,NP)
      RETURN
 101  FORMAT(2I4)
 102  FORMAT(E20.10)
C END SUBROUTINE FARRAY
      END
      SUBROUTINE TFRMTX(X1,X2,NR,NC,ITX)
      COMMON/MAIN1/NDIM
      DIMENSION X1(1),X2(1)
      IF(ITX.EQ.2) GO TO 20
      J=NC*NDIM
      KK=0
      DO 10 I=1,J,NDIM
      L=I+NR-1
      DO 10 JJ=I,L
      KK=KK+1
 10   X1(KK)=X2(JJ)
      RETURN
 20   KK=NR*NC+1
      DO 30 I=1,NC
      L=(NC-I)*NDIM+1
      DO 30 J=1,NR
      KK=KK-1
      JJ=L+NR-J
 30   X2(JJ)=X1(KK)
      RETURN
      END
      SUBROUTINE MATLST(A,NR,NC,NT,KDEV)
      DIMENSION A(NR,NC)
      WRITE(KDEV,101) NT
      DO 10 I=1,NR
 10   WRITE(KDEV,102) (A(I,J),J=1,NC)
 101  FORMAT(1H0,A3," MATRIX"/)
 102  FORMAT(1X1P10G13.4)
      RETURN
C END SUBROUTINE MATLST
      END
      SUBROUTINE NDSCRT(A,N,NTERMS)
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      DIMENSION A(1)
      NTERMS=IFIX(6.+3.*TSAMP*XNORM(N,A))
      IF(NTERMS.GT.30) NTERMS=30
      RETURN
C END SUBROUTINE NDSCRT
      END
      SUBROUTINE RQWGTS(W,ND,NP)
      DIMENSION W(1)
 10   WRITE 101
 101  FORMAT(" ENTER I AND QW(I,I)--(0/ WHEN COMPLETE) >")
 15   READ*,I,V
      IF(I.EQ.0) RETURN
      IF(I.LE.ND) GO TO 20
      WRITE 102
 102  FORMAT(" ERROR IN ARRAY INDEX")
      GO TO 10
 20   IF(V) 25,30,40
 25   WRITE 103
 103  FORMAT(" ELEMENTS MUST BE NON-NEGATIVE")
      GO TO 15
 30   IF(NP) 35,40,35
 35   WRITE 104
 104  FORMAT(" ELEMENTS MUST BE POSITIVE")
      GO TO 15
 40   L1=LADDR(ND,I,I)
      W(L1)=V
      GO TO 15
C END SUBROUTINE RQWGTS
      END
      SUBROUTINE DVCTOR(N,A,V)
      DIMENSION A(1),V(1)
      NP1=N+1
      N2=N*N
      J=0
      DO 10 I=1,N2,NP1
      J=J+1
 10   V(J)=A(I)
      RETURN
```

```
C END SUBROUTINE DVCTOR
      END
      SUBROUTINE POLES(A,N,ITYPE,ZM1,ZM2)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      DIMENSION NTYP(5),A(1),ZM1(1),ZM2(1)
      DATA NTYP/6HDESIGN,7HCOMMAND,5HTRUTH,5HREGPI,6HFILTER/
      NDS=NDIM
      NDIM=N
      NDIM1=NDIM+1
      CALL EIGEN(NDIM,A,ZM1,ZM1(NDIM1),ZM2,0)
      IF(ITYPE.LT.4) GO TO 10
      CALL MAPOLE(N,ZM1,ZM1(NDIM1),TSAMP)
 10   WRITE(KLIST,102) NTYP(ITYPE)
      WRITE(KTERM,102) NTYP(ITYPE)
      WRITE(KLIST,101) (ZM1(I),ZM1(NDIM+I),I=1,N)
      WRITE(KTERM,101) (ZM1(I),ZM1(NDIM+I),I=1,N)
      NDIM=NDS
      NDIM1=NDIM+1
 101  FORMAT(6X,1PE15.7,"  +J(",1PE15.7,")")
 102  FORMAT("0POLES OF ",A7," MATRIX"/)
      RETURN
C END SUBROUTINE POLES
      END
      SUBROUTINE MAPOLE(N,ZR,ZI,T)
      DIMENSION ZR(1),ZI(1)
      RT=1./T
      DO 10 I=1,N
      ZM=SQRT(ZR(I)**2+ZI(I)**2)
      SIGMA=RT*ALOG(ZM)
      ZI(I)=RT*ATAN2(ZI(I),ZR(I))
 10   ZR(I)=SIGMA
      RETURN
C END SUBROUTINE MAPOLE
      END
      FUNCTION LADDR(NR,I,J)
      LADDR=I+NR*(J-1)
      RETURN
C END FUNCTION LADDR
      END
      SUBROUTINE FTMTX(X,Y,NR,NC)
      DIMENSION X(1),Y(1)
      NE=NR*NC
      DO 10 I=1,NE
 10   Y(I)=X(I)
      RETURN
C END SUBROUTINE FTMTX
      END

      SUBROUTINE FMMUL(X,Y,NR1,NC1,NC2,Z)
      DIMENSION X(NR1,NC1),Y(NC1,NC2),Z(NR1,NC2)
      DOUBLE PRECISION TD
      DO 10 I=1,NR1
      DO 10 J=1,NC2
      TD=0.D00
      DO 5 K=1,NC1
 5    TD=TD+X(I,K)*Y(K,J)
 10   Z(I,J)=TD
      RETURN
C END SUBROUTINE FMMUL
      END
      SUBROUTINE FTMUL(X,Y,NR1,NC1,NC2,Z)
      DIMENSION X(NR1,NC1),Y(NR1,NC2),Z(NC1,NC2)
      DOUBLE PRECISION TD
      DO 10 I=1,NC1
      DO 10 J=1,NC2
      TD=0.D00
      DO 5 K=1,NR1
 5    TD=TD+X(K,I)*Y(K,J)
 10   Z(I,J)=TD
      RETURN
C END SUBROUTINE FTMUL
      END
      SUBROUTINE FMADD(X,Y,NR,NC,Z)
      DIMENSION X(1),Y(1),Z(1)
      NE=NR*NC
      DO 10 I=1,NE
 10   Z(I)=X(I)+Y(I)
      RETURN
C END SUBROUTINE FMADD
      END
      SUBROUTINE ZPART(A,NR,NC,ND)
      DIMENSION A(1)
      NE=NC*ND
      DO 10 I=1,NR
      DO 10 J=I,NE,ND
 10   A(J)=0.
      RETURN
C END SUBROUTINE ZPART
      END
      SUBROUTINE SUBI(A,NR,ND)
      DIMENSION A(1)
      ND1=ND+1
      NE=NR*ND
      DO 10 I=1,NE,ND1
 10   A(I)=A(I)-1.
      RETURN
C END SUBROUTINE SUBI
      END
      SUBROUTINE WPLOTF(V,N)
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      DIMENSION V(N)
      WRITE(KPLOT,101) (V(I),I=1,N)
      RETURN
 101  FORMAT(E20.10)
C END SUBROUTINE WPLOTF
      END
      SUBROUTINE RPLOTF(V,N,IERR)
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      DIMENSION V(N)
      READ(KPLOT,101) (V(I),I=1,N)
      IF(EOF(KPLOT)) 5,10
5     IERR=1
      RETURN
10    IERR=0
      RETURN
101   FORMAT(E20.10)
C END SUBROUTINE RPLOTF
      END
      SUBROUTINE STRPLT(A,V,NS,NV,NP,NVO)
      DIMENSION A(1),NS(1),NV(1),V(1)
      A(NS(1))=V(NVO)
      DO 5 I=1,NP
5     A(NS(I+1))=V(NV(I))
      RETURN
C END SUBROUTINE STRPLT
      END
      SUBROUTINE PLOTLP(N,M,A,IPSC,ISCL,LPTERM,NDEV,ITITLE)
C * * * * * * * * * * *
C * N = NUMBER OF POINTS TO BE PLOTTED
C * M = NUMBER OF OUTPUTS TO BE PLOTTED
C * A = VECTOR OF SAMPLE POINTS FOR PLOTTING : DIMENSION = N*M
C *     ELEMENTS 1 TO N ARE THE INDEPENDENT VARIABLE
C *     ELEMENTS (N+1) TO 2*N, (2*N+1) TO 3*N, AND SO ON ARE
C *     THE DEPENDENT VARIABLES--EACH VARIABLE IS IN CONSECUTIVE
C *     STORAGE WITH CORRESPONDING SAMPLE POINTS FOR EACH
C *     SEPARATED BY MULTIPLES OF N
C * IPSC = -1 => SCALE ALL VARIABLES TOGETHER (1 PLOT)
C *      =  0 => SCALE TOGETHER AND SEPARATELY (2 PLOTS)
C *      = +1 => SCALE SEPARATELY (1 PLOT)
C * ISCL = 0 => PLOT OVER EXACT RANGE OF VARIABLE
C *      = 1 => PLOT USING EVEN SCALING
C * LPTERM = 0 => PLOT IS TO TERMINAL (50 CHARACTERS WIDE)
C *        = 1 => PLOT IS TO LINE PRINTER (100 CHARACTERS WIDE)
C * NDEV = DEVICE NUMBER FOR PLOT OUTPUT
C * ITITLE = VECTOR(DIMENSIONED 5) WITH 50 CHARACTER TITLE
C * * * * * * * * * * *
      DIMENSION YSCAL(6),YMIN(6),IBLNK(6),YPR(11),A(1)
      INTEGER OUT(101),SYMBOL(6),BLANK,PLUS,GRID,ITITLE(5)
      DATA BLANK,PLUS,COLON,SYMBOL/1H ,1H+,1H:,1H1,1H2,1H3,1H4,1H5,1H6/
1     FORMAT(1H )
2     FORMAT(1H1,11X,5A10/)
10    FORMAT(1H ,F11.2,6X,101A1)
12    FORMAT(11H0    SCALE ,A1,1X,11F10.4)
      IPAPER=5*(1+LPTERM)
      ISPAC=10*IPAPER
      RISPAC=FLOAT(ISPAC)
      ISPAC=ISPAC+1
      IPRT1=IPAPER+1
      RMIN=A(N+1)
      RMAX=RMIN
25    DO 41 ISC=1,M
      M1=ISC*N+1
      YL=A(M1)
      YH=YL
      M2=N*(ISC+1)
      DO 40 J=M1,M2
      IF(A(J).LT.YL) GO TO 30
      IF(A(J).GT.YH) YH=A(J)
      GO TO 40
   30 YL=A(J)
   40 CONTINUE
      IF(YL.LT.RMIN)RMIN=YL
      IF(YH.GT.RMAX) RMAX=YH
      IF(IPSC.GE.0) CALL VARSCL(YL,YH,YSCAL(ISC),RISPAC,ISCL)
   41 YMIN(ISC)=YL
      IF(IPSC.LE.0)CALL VARSCL(RMIN,RMAX,SCAL,RISPAC,ISCL)
      IC=2-IABS(IPSC)
      DO 45 IX=1,ISPAC
   45 OUT(IX)=BLANK
      DO 100 ICO=1,IC
      WRITE(NDEV,2) (ITITLE(I),I=1,5)
      DO 60 I=1,N
      XPR=A(I)
      IF(MOD(I,10).EQ.0) GO TO 458
      GRID=BLANK
      GO TO 460
  458 GRID=COLON
  460 DO 461 IX=2,ISPAC,2
  461 OUT(IX)=GRID
      DO 46 IX=1,ISPAC,10
   46 OUT(IX)=PLUS
      DO 55 J=1,M
      IL=I+J*N
      IF(IPSC) 48,47,49
   47 IPSCT=IPSC+ICO
      IF(IPSCT.EQ.2) GO TO 49
   48 JP=IFIX((A(IL)-RMIN)/SCAL)+1
      GO TO 50
   49 JP=IFIX((A(IL)-YMIN(J))/YSCAL(J))+1
   50 OUT(JP)=SYMBOL(J)
   55 IBLNK(J)=JP
      WRITE(NDEV,10) XPR,(OUT(IX),IX=1,ISPAC)
      DO 59 J=1,M
      ITEMP=IBLNK(J)
   59 OUT(ITEMP)=BLANK
   60 CONTINUE
      IF(IPSC) 68,67,72
   67 IF(IPSCT.EQ.2) GO TO 72
   68 YPR(1)=RMIN
      DO 70 I=1,IPAPER
   70 YPR(I+1)=YPR(I)+10.*SCAL
      WRITE(NDEV,12) BLANK,(YPR(I),I=1,IPRT1)
      GO TO 100
   72 DO 76 ISC=1,M
      YPR(1)=YMIN(ISC)
      DO 74 I=1,IPAPER
   74 YPR(I+1)=YPR(I)+10.*YSCAL(ISC)
   76 WRITE(NDEV,12) SYMBOL(ISC),(YPR(IX),IX=1,IPRT1)
  100 WRITE(NDEV,1)
      RETURN
C END SUBROUTINE PLOTLP
      END
      SUBROUTINE VARSCL(XMIN,XMAX,SCALE,RSPACE,ISCL)
      IF(XMAX.EQ.XMIN) XMIN=.9*XMIN-10.
      SCALE=XMAX-XMIN
      IF(ISCL.EQ.0) GO TO 25
      EXP=IFIX(100.+ALOG10(SCALE))-100.
      FACTOR=10.**(1.-EXP)
      XMINT=XMIN*FACTOR
      XMAXT=XMAX*FACTOR
      IF(XMAXT.GE.0.) XMAXT=XMAXT+.9
      IF(XMINT.LE.0.) XMINT=XMINT-.9
      XMINT=AINT(XMINT)
      ISCAL=XMAXT-XMINT
      IF(MOD(ISCAL,5).NE.0) ISCAL=ISCAL+5-MOD(ISCAL,5)
      FACTOR=10.**(EXP-1.)
      XMIN=XMINT*FACTOR
      SCALE=FACTOR*FLOAT(ISCAL)
   25 SCALE=SCALE/RSPACE
      RETURN
C END SUBROUTINE VARSCL
      END
      SUBROUTINE PFDATA(ICODE,ND)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/MAIN2/COM2(1)
      COMMON/INOU/KIN,KOUT,KPUNCH
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/SYSMTX/NVSM,SM(1)
      COMMON/ZMTX1/NVZM,ZM1(1)
      COMMON/ZMTX2/ZM2(1)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/LOCD/LAP,LGP,LPHI,LBD,LEX,LPHD,LQ,LQN,LQD,LC,LDY,LEY,LHP,LR
      COMMON/DSNMTX/NVDM,NODY,NOEY,DM(1)
      COMMON/NDIMC/NNC,NRC,NPC
      COMMON/LOCC/LPHC,LBDC,LCC,LDC
      COMMON/CMDMTX/NVCM,NEWCM,NODC,CM(1)
      COMMON/NDIMT/NNT,NRT,NMT,NWT
      COMMON/LOCT/LPHT,LBDT,LQDT,LHT,LRT,LTDT,LTNT
      COMMON/TRUMTX/NVTM,TM(1)
      COMMON/LCNTRL/LPI11,LPI12,LPI21,LPI22,LPHDL,LBDL
      COMMON/CONTROL/NVCTL,CTL(1)
      COMMON/LREGPI/LXDW,LUDW,LPHCL,LGC1,LGC2,LEL,LEE
      COMMON/CREGPI/NVRPI,RPI(1)
      COMMON/LCGT/LA11,LA13,LA21,LA23,LA12,LA22,LELA11,LELA12,LELA13
      COMMON/CCGT/NVCGT,CGT(1)
      COMMON/LKF/LEADSN,LFLTRK,LFCOV
      COMMON/CKF/NVFLT,FLT(1)
      DIMENSION ND(1)
      ND(1)=NND
      ND(2)=NRD
      ND(3)=NPD
      ND(4)=NMD
      ND(5)=NDD
      ND(6)=NPLD
      ND(7)=NNC
      ND(8)=NRC
      ND(9)=NPC
      ND(10)=NNT
      ND(11)=NRT
      ND(12)=NMT
      ND(13)=NODY
      ND(14)=NOEY
      NVZMS=NVZM
      CALL FTMTX(DM(LPHI),SM,NND,NND)
      LL=NND*NND+1
      CALL FTMTX(DM(LBD),SM(LL),NND,NRD)
      LL=NND*NRD+LL
      IF(NDD.EQ.0) GO TO 100
            CALL FTMTX(DM(LEX),SM(LL),NND,NDD)
            LL=NND*NDD+LL
            CALL FTMTX(DM(LPHD),SM(LL),NDD,NDD)
            LL=NDD*NDD+LL
            IF(NODY.EQ.1) GO TO 90
                  CALL FTMTX(DM(LDY),SM(LL),NPD,NRD)
                  LL=NPD*NRD+LL
 90         IF(NOEY.EQ.1) GO TO 95
                  CALL FTMTX(DM(LEY),SM(LL),NPD,NDD)
                  LL=NPD*NDD+LL
 95         CALL FTMTX(DM(LHP),SM(LL),NMD,NPLD)
            LL=NMD*NPLD+LL
            GO TO 200
 100  CONTINUE
      IF(NODY.EQ.1) GO TO 105
            CALL FTMTX(DM(LDY),SM(LL),NPD,NRD)
            LL=NPD*NRD+LL
 105  CALL FTMTX(DM(LHP),SM(LL),NMD,NND)
      LL=NMD*NND+LL
 200  CALL FTMTX(DM(LC),SM(LL),NPD,NND)
      LL=NPD*NND+LL
      CALL FTMTX(CM(LPHC),SM(LL),NNC,NNC)
      LL=NNC*NNC+LL
      CALL FTMTX(CM(LBDC),SM(LL),NNC,NRC)
      LL=NNC*NRC+LL
      CALL FTMTX(CM(LCC),SM(LL),NPC,NNC)
      LL=NPC*NNC+LL
      CALL FTMTX(TM(LPHT),SM(LL),NNT,NNT)
      LL=NNT*NNT+LL
      CALL FTMTX(TM(LBDT),SM(LL),NNT,NRT)
      LL=NNT*NRT+LL
      CALL FTMTX(TM(LQDT),SM(LL),NNT,NNT)
      LL=NNT*NNT+LL
      CALL FTMTX(TM(LHT),SM(LL),NMT,NNT)
      LL=NMT*NNT+LL
      CALL FTMTX(TM(LRT),SM(LL),NMT,NMT)
      LL=NMT*NMT+LL
      CALL FTMTX(RPI(LGC1),SM(LL),NRD,NND)
      LL=NRD*NND+LL
      CALL FTMTX(RPI(LGC2),SM(LL),NRD,NPD)
      LL=NRD*NPD+LL
      CALL FTMTX(CGT(LELA11),SM(LL),NRC,NNC)
      LL=NRC*NNC+LL
      IF(NDD.EQ.0) GO TO 300
           CALL FTMTX(CGT(LELA13),SM(LL),NRD,NDD)
           LL=NRD*NDD+LL
           CALL FTMTX(FLT(LFLTRK),SM(LL),NPLD,NMD)
           LL=NPLD*NMD+LL
           CALL FTMTX(TM(LTDT),SM(LL),NND,NNT)
           LL=NND*NNT+LL
           CALL FTMTX(TM(LTNT),SM(LL),NDD,NNT)
           LL=NDD*NNT+LL
           GO TO 310
  300 CONTINUE
      CALL FTMTX(FLT(LFLTRK),SM(LL),NND,NMD)
      LL=NND*NMD+LL
      CALL FTMTX(TM(LTDT),SM(LL),NND,NNT)
      LL=NND*NNT+LL
  310 SM(LL)=TSAMP
      ND(15)=LL
      CALL WFILED(ICODE,LL,ND,SM)
      NVZM=NVZMS
      RETURN
C END SUBROUTINE PFDATA
      END
      SUBROUTINE IMPLEX(A)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/SYSMTX/NVSM,SM(1)
      COMMON/ZMTX1/NVZM,ZM1(1)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/LOCD/LAP,LGP,LPHI,LBD,LEX,LPHD,LQ,LQN,LQD,LC,LDY,LEY,LHP,LR
      COMMON/DSNMTX/NVDM,NODY,NOEY,DM(1)
      COMMON/NDIMC/NNC,NRC,NPC
      COMMON/AMC/AM(1)
      COMMON/BDG/BD(1)
      DIMENSION A(1)
      DATA NO/1HN/
  215 WRITE(KLIST,126)
  126 FORMAT(5("* "),"COMBINED IMPLICIT/EXPLICIT CONTROL",5(" *"))
      LQI=1
      LRII=LQI+NPD*NPD
      LRRI=LRII+NRD*NRD
      NPDNRD=NPD+NRD
  120 CALL ZPART(A,NPDNRD,NPDNRD,NPDNRD)
      WRITE 122,NPD
  122 FORMAT(" ENTER WEIGHTS ON IMPLICIT OUTPUT DERIVATIVES: ",I2)
      CALL RQWGTS(A,NPD,0)
      WRITE 124,NRD
  124 FORMAT(" ENTER WEIGHTS ON IMPLICIT CONTROL MAGNITUDES: ",I2)
      CALL RQWGTS(A(LRII),NRD,1)
      L1=1
      L2=L1+NND*NND
      L3=L2+NPD*NND
      L4=L3+NPD*NND
      L5=L4+NPD*NND
      L6=L5+NND*NPD
      L7=L6+NPD*NND
      IF(NDD.EQ.0) GO TO 5
           NDIM=NPLD
           NDIM1=NDIM+1
           GO TO 10
 5    NDIM=NND
      NDIM1=NDIM+1
 10   CALL TFRMTX(SM(L1),DM(LAP),NND,NND,1)
      CALL FMMUL(DM(LC),SM(L1),NPD,NND,NND,SM(L2))
      CALL FMMUL(AM,DM(LC),NPD,NPD,NND,SM(L3))
      NDIM=NPD
      NDIM1=NDIM+1
      CALL MADD1(NPD,NND,SM(L2),SM(L3),SM(L4),-1.)
      CALL FTRNSP(NPD,NND,SM(L4),SM(L5))
      CALL FMMUL(A,SM(L4),NPD,NPD,NND,SM(L6))
      CALL FMMUL(SM(L5),SM(L6),NND,NPD,NND,SM(L1))
      CALL FTRNSP(NND,NRD,BD,SM(L2))
      CALL FTRNSP(NPD,NND,DM(LC),SM(L3))
      CALL FMMUL(SM(L2),SM(L3),NRD,NND,NPD,SM(L4))
      CALL FMMUL(SM(L4),SM(L6),NRD,NPD,NND,SM(L5))
      CALL FTMTX(SM(L5),SM(L2),NRD,NND)
      CALL FMMUL(SM(L4),A,NRD,NPD,NPD,SM(L5))
      CALL FMMUL(SM(L5),DM(LC),NRD,NPD,NND,SM(L6))
      CALL FMMUL(SM(L6),BD,NRD,NND,NRD,SM(L7))
      CALL MADD1(NRD,NRD,A(LRII),SM(L7),SM(L5),1.)
      CALL FTMTX(SM(L5),SM(L3),NRD,NRD)
      WRITE 400
 400  FORMAT("OLIST 'QIH' MATRIX TO TERMINAL (Y OR N) >")
      READ 410,IANS
 410  FORMAT(A3)
      IF (IANS.EQ.NO)GO TO 490
      CALL MATLST(SM(L1),NND,NND,"QIH",KTERM)
      WRITE 420
 420  FORMAT("OCHANGE IMPLICIT WEIGHTS (Y OR N) >")
      READ 410,IANS
      IF(IANS.EQ.NO)GO TO 490
      GO TO 120
 490  CALL MATLST(A,NPD,NPD,"QI",KLIST)
      CALL DVCTOR(NPD,A,A(LRRI))
      CALL MATLST(A(LRRI),NPD,1,"QI",KTERM)
      CALL MATLST(A(LRII),NRD,NRD,"RI",KLIST)
      CALL DVCTOR(NRD,A(LRII),A(LRRI))
      CALL MATLST(A(LRRI),NRD,1,"RI",KTERM)
      CALL MATLST(SM(L1),NND,NND,"QIH",KLIST)
      CALL MATLST(SM(L2),NRD,NND,"SIH",KLIST)
      CALL MATLST(SM(L3),NRD,NRD,"RIH",KLIST)
      RETURN
C END SUBROUTINE IMPLEX
      END
      SUBROUTINE MODXU(X,U)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/MAIN2/COM2(1)
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/SYSMTX/NVSM,SM(1)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/NDIMC/NNC,NRC,NPC
      DIMENSION X(1),U(1)
      L1=1
      L2=L1+NND*NND
      L3=L2+NRD*NND
      L4=L3+NRD*NRD
      L5=L4+NND*NND
      L6=L5+NND*NND
      NDIM=NNPR
      NDIM1=NDIM+1
      CALL TFRMTX(SM(L4),X,NND,NND,1)
      CALL FMADD(SM(L4),SM,NND,NND,SM(L5))
      CALL TFRMTX(SM(L5),X,NND,NND,2)
      DO 10 I=1,NRD
         LA=NNPR*(I-1)
         LB=L2+I-1
         LE=LB+NRD*(NND-1)
         DO 10 J=LB,LE,NRD
         LA=LA+1
   10 COM2(LA)=COM2(LA)+SM(J)
      CALL FMADD(SM(L3),U,NRD,NRD,U)
      CALL MATLST(X,NNPR,NNPR,"XIE",KLIST)
      RETURN
C END SUBROUTINE MODXU
      END
      SUBROUTINE FTRNSP(NR,NC,A,B)
      DIMENSION A(1),B(1)
      DO 10 I=1,NR
           DO 20 J=1,NC
                B((I-1)*NC+J)=A((J-1)*NR+I)
   20 CONTINUE
   10 CONTINUE
      RETURN
C END SUBROUTINE FTRNSP
      END
*EOR
```

C.2 Subroutine Changes for CGT/PI Formulation 2

Subroutines CGTXQ, SREGPI, SCGT, and CGTKX were changed to produce the program based on CGT/PI formulation 2. These subroutines are listed on the following pages. Changed lines are marked.

```
      SUBROUTINE CGTXQ
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/MAIN2/COM2(1)
      COMMON/INOU/KIN,KOUT,KPUNCH
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/SYSMTX/NVSM,SM(1)
      COMMON/ZMTX1/NVZM,ZM1(1)
      COMMON/ZMTX2/ZM2(1)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/LOCD/LAP,LGP,LPHI,LBD,LEX,LPHD,LQ,LQN,LQD,LC,LDY,LEY,LHP,LR
      COMMON/DSNMTX/NVDM,NODY,NOEY,DM(1)
      COMMON/NDIMC/NNC,NRC,NPC
      COMMON/LOCC/LPHC,LBDC,LCC,LDC
      COMMON/CMDMTX/NVCM,NEWCM,NODC,CM(1)
      COMMON/NDIMT/NNT,NRT,NMT,NWT
      COMMON/LOCT/LPHT,LBDT,LQDT,LHT,LRT,LTDT,LTNT
      COMMON/TRUMTX/NVTM,TM(1)
      COMMON/LCNTRL/LPI11,LPI12,LPI21,LPI22,LPHDL,LBDL
      COMMON/CONTROL/NVCTL,CTL(1)
      COMMON/LREGPI/LXDW,LUDW,LPHCL,LGC1,LGC2,LEL,LEE
      COMMON/CREGPI/NVRPI,RPI(1)
      COMMON/LCGT/LA11,LA13,LA21,LA23,LA12,LA22,LELA11,LELA12,LELA13
      COMMON/CCGT/NVCGT,CGT(1)
      COMMON/LKF/LEADSN,LFLTRK,LFCOV
      COMMON/CKF/NVFLT,FLT(1)
      COMMON/AMC/AM(1)
      COMMON/BDG/BD(1)
      DIMENSION LD(15),ND(15)
      DATA NPLTZM/606/
      DATA IEOI,NO/-1,1HN/
      REWIND KLIST
      WRITE(KLIST,115) DATE(DUM),TIME(DUM)
      WRITE(KTERM,115) DATE(DUM),TIME(DUM)
  115 FORMAT("1",27X,"* * * CGTPIQ * * *"/14X,
     1 "PROGRAM TO DESIGN A COMMAND GENERATOR TRACKER"/8X,
     2 "USING A REGULATOR WITH PROPORTIONAL PLUS INTEGRAL CONTROL"/13X,
     3 "BASED ON THE INTEGRAL OF THE REGULATION ERROR,"/16X,
     4 "AND A KALMAN FILTER FOR STATE ESTIMATION."/28X,
     5 "* * * CGTPIQ * * *"//11X,"DATE : ",A10//,11X,
     6 "TIME : ",A10////)
      REWIND KSAVE
      REWIND KDATA
      WRITE(KSAVE,112) IEOI,NPLTZM
      DO 10 I=1,15
   10 ND(I)=0
      DO 12 I=1,15
   12 LD(I)=1
      LFLRPI=0
      LFLCGT=0
      LFLKF=0
      LTEVAL=0
      LABORT=0
      IPI=0
      ICGT=0
      ITRU=0
      IFLTR=0
      ICODE=4
      LFAVAL=0
      LGCGT=0
      NVCOM=MINO(NDIM,NVZM)
      KOUT=KLIST
      KPUNCH=KPLOT
      IF(NVSM.GE.NPLTZM) GO TO 50
      WRITE 101,NPLTZM
      GO TO 1000
50    WRITE 102
      READ*,TSAMP
      IF(TSAMP.LE.0.) GO TO 50
      WRITE(KLIST,103) TSAMP
103   FORMAT("0SAMPLE PERIOD IS ",F5.3," SECONDS")
      CALL SETUP(ND,LD,ICGT,ITRU,1)
      IF(LABORT) 1000,100,1000
100   LABORT=0
      IMPLIC=0
      WRITE 104
104   FORMAT("0CONTROLLER DESIGN (Y OR N) >")
      READ 111,IANS
      IF(IANS.EQ.NO) GO TO 500
      LFLKF=0
      CALL PIMTX(IPI)
      IF(LABORT) 1000,125,1000
125   WRITE 105
105   FORMAT("0DESIGN REG/PI (Y OR N) >")
      READ 111,IANS
      IF(IANS.EQ.NO) GO TO 150
      CALL SETUP(ND,LD,ICGT,ITRU,4)
      IF(ICGT.EQ.0) GO TO 125
      WRITE 400
400   FORMAT("0INCORPORATE IMPLICIT MODEL (Y OR N) >")
      READ 111,IANS
      IF(IANS.EQ.NO) GO TO 490
      IMPLIC=1
      CALL SETUP(ND,LD,ICGT,ITRU,4)
      IF(ICGT.NE.0) GO TO 460
      IMPLIC=0
      GO TO 480
460   IF(NPD.EQ.NNC) GO TO 480
      WRITE 470
470   FORMAT("0COMMAND MODEL STATE DIM MUST EQUAL SYSTEM OUTPUT DIM")
      LABORT=-1
480   IF(LABORT) 100,490,1000
490   CALL SREGPI(IMPLIC)
      IF(LABORT) 1000,200,1000
150   WRITE 106
106   FORMAT("0DESIGN CGT (Y OR N) >")
      READ 111,IANS
      IF(IANS.EQ.NO) GO TO 100
      CALL SETUP(ND,LD,ICGT,ITRU,2)
      IF(ICGT) 155,100,155
  155 IF(LABORT) 100,160,1000
  160 CALL SCGT
      IF(LABORT) 100,170,1000
  170 IF(LFLCGT.LE.0) GO TO 125
  200 LABORT=0
      WRITE 107
  107 FORMAT("0CONTROLLER EVALUATION WRT TRUTH MODEL (Y OR N) >")
      READ 111,IANS
      IF(IANS.EQ.NO) GO TO 250
      CALL SETUP(ND,LD,ICGT,ITRU,3)
      IF(LABORT) 200,260,1000
  250 LTEVAL=0
  260 CALL CEVAL
      IF(LFLCGT.EQ.1) LGCGT=1
      IF(LFAVAL.EQ.0.OR.LGCGT.EQ.0) GO TO 100
  270 WRITE 600
  600 FORMAT("0WRITE PERFORMANCE EVALUATION DATA TO 'SAVE' FILE (Y OR N)
     +>")
      READ 111,IANS
      IF(IANS.EQ.NO) GO TO 100
      ICODE=ICODE+1
      CALL PFDATA(ICODE,ND)
      INUM=ICODE-4
      WRITE 605,INUM
  605 FORMAT("0PERFORMANCE EVALUATION DATA, NO. "I2,",WRITTEN TO 'SAVE
     +' FILE")
      GO TO 100
  500 LABORT=0
      WRITE 108
  108 FORMAT("0FILTER DESIGN (Y OR N) >")
      READ 111,IANS
      IF(IANS.EQ.NO) GO TO 900
      CALL FLTRK(IFLTR)
      IF(IFLTR.EQ.0) GO TO 900
      IF(LABORT) 1000,510,1000
  510 CALL SETUP(ND,LD,ICGT,ITRU,3)
      IF(LABORT) 500,525,1000
  525 CALL FEVAL
  530  IF(LABORT) 1000,540,1000
  540  LFAVAL=1
      IF(LGCGT.EQ.1) GO TO 270
      GO TO 500
  900 WRITE 109
  109 FORMAT("0END DESIGN RUNS (Y OR N) >")
      READ 111,IANS
      IF(IANS.EQ.NO) GO TO 100
      IF(LFLRPI.EQ.0) GO TO 1000
      NPNTS=NRD*(NNPR+NND)
      ND(1)=NPNTS
      ND(2)=LGC1
      ND(3)=LGC2
      ND(4)=LEL
      CALL WFILED(4,NPNTS,ND,RPI(LGC1))
      WRITE 113
```

```
 1000 CONTINUE
      WRITE(KLIST,110)
      REWIND KSAVE
      REWIND KDATA
      REWIND KLIST
      WRITE 110
 101  FORMAT("0INSUFFICIENT MEMORY /SYSMTX/, NEED: ",I4)
 102  FORMAT("0ENTER SAMPLE PERIOD FOR DIGITAL CONTROLLER >")
 110  FORMAT("0PROGRAM EXECUTION STOP")
 111  FORMAT(A3)
 112  FORMAT(2I4)
 113  FORMAT(6X,"REG/PI GAINS WRITTEN TO 'SAVE' FILE")
      RETURN
C END SUBROUTINE CGTXQ
      END



      SUBROUTINE SREGPI(IMPLIC)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/SYSMTX/NVSM,SM(1)
      COMMON/ZMTX1/NVZM,ZM1(1)
      COMMON/ZMTX2/ZM2(1)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/LCNTRL/LPI11,LPI12,LPI21,LPI22,LPHDL,LBDL
      COMMON/CONTROL/NVCTL,CTL(1)
      COMMON/LREGPI/LXDW,LUDW,LPHCL,LGC1,LGC2,LEL,LEE
      COMMON/CREGPI/NVRPI,RPI(1)
      COMMON/LOCD/LAP,LGP,LPHI,LBD,LEX,LPHD,LQ,LQN,LQD,LC,LDY,LEY,LHP,LR
      COMMON/DSNMTX/NVDM,NODY,NOEY,DM(1)
      COMMON/NDIMC/NNC,NRC,NPC
      COMMON/LOCC/LPHC,LBDC,LCC,LDC
      COMMON/CMDMTX/NVCM,NEWCM,NODC,CM(1)
      WRITE(KLIST,110)
 110  FORMAT(////11X,5("* "),"REG/PI DESIGN",5(" *")////)
      NSIZE=NRD*(5*NRD+2*NND)+NNPR*NNPR
      IF(NSIZE.LE.NVRPI) GO TO 5
      WRITE 101,NSIZE
 101  FORMAT("0INSUFFICIENT MEMORY /CREGPI/, NEED: ",I4)
      GO TO 8
 5    NSIZE=NNPR*(3*NNPR+NRD)
      IF(NSIZE.LE.NVSM) GO TO 10
      WRITE 102,NSIZE
 102  FORMAT("0INSUFFICIENT MEMORY /SYSMTX/, NEED: ",I4)
 8    LABORT=NSIZE
      RETURN
 10   LX=1
      LU=LX+NNPR*NNPR
      CALL WXUS(SM(LX),SM(LU),COM1,ZM1,ZM2,IMPLIC)
      LUIST=LU+NNPR*NRD
      LPHP=LUIST+NNPR*NNPR
      CALL PXUP(CTL(LPHDL),CTL(LBDL),SM(LX),SM(LU),COM1,ZM2,
     1 SM(LUIST),SM(LPHP),SM(LX),ZM1)
```

```
      CALL DRIC(NDIM,SM(LPHP),ZM2,SM(LX),ZM1,RPI(LPHCL))
      CALL GCSTAR(SM(LPHP),CTL(LBDL),SM(LU),ZM1,SM(LUIST),SM(LX),ZM2)
      CALL TFRMTX(RPI(LGC1),SM(LX),NRD,NND,1)
      L3=LADDR(NDIM,1,NND+1)+LX-1
      CALL TFRMTX(RPI(LGC2),SM(L3),NRD,NRD,1)
      L1=LADDR(NNPR,NND+1,NND+1)
      L2=LADDR(NNPR,1,NND+1)
      CALL GMINV(NRD,NRD,ZM1(L1),ZM1(L2),MR,1)
      L1=LADDR(NNPR,NND+1,1)
      CALL MMUL(ZM1(L2),ZM1(L1),NRD,NRD,NND,ZM1)
      CALL MMUL(SM(L3),ZM1,NRD,NRD,NND,ZM1(L1))
      CALL TFRMTX(COM1,ZM1(L1),NRD,NND,1)
      CALL MADD1(NRD,NND,SM(LX),ZM1(L1),ZM1,-1.)
      CALL TFRMTX(RPI(LEL),ZM1,NRD,NND,1)
      CALL FMMUL(RPI(LEL),CTL(LPI12),NRD,NND,NRD,RPI(LEE))
      CALL FMADD(RPI(LEE),CTL(LPI22),NRD,NRD,RPI(LEE))
      CALL MATLST(RPI(LGC1),NRD,NND,"GC1",KLIST)
      CALL MATLST(RPI(LGC1),NRD,NND,"GC1",KTERM)
      CALL MATLST(RPI(LGC2),NRD,NRD,"GC2",KLIST)
      CALL MATLST(RPI(LGC2),NRD,NRD,"GC2",KTERM)
      CALL MATLST(RPI(LEE),NRD,NRD,"E",KLIST)
      IF(NODY.EQ.1)GO TO 15
      CALL FMMUL(RPI(LEE),DM(LDY),NRD,NRD,NRD,ZM1)
      CALL SUBI(ZM1,NRD,NRD)
      NR2=NRD*NRD
      DO 12 I=1,NR2
 12   ZM1(I)=-ZM1(I)
      NDIM=NRD
      NDIM1=NDIM+1
      CALL GMINV(NRD,NRD,ZM1,ZM2,MR,1)
      CALL FMMUL(ZM2,RPI(LEE),NRD,NRD,NRD,ZM1)
      CALL MATLST(ZM1,NRD,NRD,"KP",KLIST)
      CALL FMMUL(ZM2,RPI(LGC2),NRD,NRD,NRD,ZM1)
      CALL MATLST(ZM1,NRD,NRD,"KI",KLIST)
      GO TO 20
 15   WRITE(KLIST,103)
 103  FORMAT("0KP = E  AND  KI = GC2")
 20   CONTINUE
      CALL FMMUL(COM1,CTL(LPI12),NRD,NND,NRD,ZM2)
      CALL FTMTX(CM(LPHC),ZM1,NNC,NNC)
      CALL SUBI(ZM1,NNC,NNC)
      NR2=NNC*NNC
      DO 25 I=1,NR2
 25   ZM1(I)=-ZM1(I)
      NDIM=NNC
      NDIM1=NDIM+1
      CALL GMINV(NNC,NNC,ZM1,COM1,MR,1)
      CALL FMMUL(CM(LCC),COM1,NPC,NNC,NNC,ZM1)
      CALL FMMUL(ZM1,CM(LBDC),NPC,NNC,NRC,COM1)
      CALL FMADD(COM1,CM(LDC),NPC,NRC,ZM1)
      CALL FMMUL(ZM2,ZM1,NRD,NRD,NRC,RPI(LEL))
      LFLRPI=1
      LFLCGT=0
      RETURN
C END SUBROUTINE SREGPI
```

```
      END


      SUBROUTINE SCGT
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/ZMTX1/NVZM,ZM1(1)
      COMMON/ZMTX2/ZM2(1)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/NDIMC/NNC,NRC,NPC
      COMMON/CMDMTX/NVCM,NEWCM,NODC,CM(1)
      COMMON/LREGPI/LXDW,LUDW,LPHCL,LGC1,LGC2,LEL,LEE
      COMMON/CREGPI/NVRPI,RPI(1)
      COMMON/LCGT/LA11,LA13,LA21,LA23,LA12,LA22,LELA11,LELA12,LELA13
      COMMON/CCGT/NVCGT,CGT(1)
      IF(NEWCM) 20,20,15
   15 NSIZE=(NND+2*NPD)*(NNC+NRC+NDD)
      IF(NSIZE.LE.NVCGT) GO TO 16
      WRITE 106,NSIZE
      LABORT=NSIZE
      RETURN
   16 IF(NND.GE.NNC) GO TO 17
      WRITE 107
      GO TO 18
   17 IF(NND.GE.NDD) GO TO 19
      WRITE 108
   18 LABORT=-1
      RETURN
   19 NEWCM=0
      LA11=1
      LA13=LA11+NND*NNC
      LA21=LA13+NND*NDD
      LA23=LA21+NPD*NNC
      LA12=LA23+NPD*NDD
      LA22=LA12+NND*NRC
      LELA11=LA22+NPD*NRC
      LELA12=LELA11+NPD*NNC
      LELA13=LELA12+NPD*NRC
      CALL CGTA(CGT(LA11),CGT(LA13),CGT(LA21),CGT(LA23),CGT(LA12),
     1 CGT(LA22),ZM1,ZM2)
   20 CALL CGTKX(CGT(LA11),CGT(LA13),CGT(LA21),CGT(LA23),CGT(LA12),
     1 CGT(LA22),CGT(LELA11),CGT(LELA12),CGT(LELA13),RPI(LEL),RPI(LGC1))
      LFLCGT=1
  106 FORMAT("0INSUFFICIENT MEMORY /CCGT/, NEED: ",I4)
  107 FORMAT("0FEWER DESIGN MODEL THAN COMMAND MODEL STATES")
  108 FORMAT("FEWER DESIGN MODEL THAN DISTURBANCE MODEL STATES")
      RETURN
C END SUBROUTINE SCGT
      END


      SUBROUTINE CGTKX(A11,A13,A21,A23,A12,A22,RELA11,RELA12,RELA13,REL,
     1 RGC)
```

```
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/NDIMC/NNC,NRC,NPC
      DIMENSION A11(1),A13(1),A21(1),A23(1),A12(1),A22(1),
     1 RELA11(1),RELA12(1),RELA13(1),REL(1),RGC(1)
      NDIM=NRD
      NDIM1=NDIM+1
      CALL FMMUL(RGC,A11,NRD,NND,NNC,RELA11)
      CALL MADD1(NRD,NNC,RELA11,A21,RELA11,1.)
      CALL MATLST(RELA11,NRD,NNC,"KXM",KLIST)
      CALL MATLST(RELA11,NRD,NNC,"KXM",KTERM)
      CALL FMMUL(RGC,A12,NRD,NND,NRC,RELA12)
      CALL MADD1(NRD,NRC,RELA12,A22,RELA12,1.)
      CALL MADD1(NRD,NRC,RELA12,REL,RELA12,-1.)
      CALL MATLST(RELA12,NRD,NRC,"KXU",KLIST)
      CALL MATLST(RELA12,NRD,NRC,"KXU",KTERM)
      IF(NDD.LT.1) RETURN
      CALL FMMUL(RGC,A13,NRD,NND,NDD,RELA13)
      CALL MADD1(NRD,NDD,RELA13,A23,RELA13,1.)
      CALL MATLST(RELA13,NRD,NDD,"KXN",KLIST)
      CALL MATLST(RELA13,NRD,NDD,"KXN",KTERM)
      RETURN
C END SUBROUTINE CGTKX
      END
```

C.3 Subroutine Changes for CGT/PI Formulation 3

Subroutines CTRESP, UCGT, SREGPI, SCGT, and CGTKX were changed to produce the program based on CGT/PI formulation 3. These subroutines are listed on the following pages. Changed lines are marked.

```
      SUBROUTINE SREGPI(IMPLIC)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/SYSMTX/NVSM,SM(1)
      COMMON/ZMTX1/NVZM,ZM1(1)
      COMMON/ZMTX2/ZM2(1)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/LCNTRL/LPI11,LPI12,LPI21,LPI22,LPHDL,LBDL
      COMMON/CONTROL/NVCTL,CTL(1)
      COMMON/LREGPI/LXDW,LUDW,LPHCL,LGC1,LGC2,LEL,LEE
      COMMON/CREGPI/NVRPI,RPI(1)
      COMMON/LOCD/LAP,LGP,LPHI,LBD,LEX,LPHD,LQ,LQN,LQD,LC,LDY,LEY,LHP,LR
      COMMON/DSNMTX/NVDM,NODY,NOEY,DM(1)
      WRITE(KLIST,110)
110   FORMAT(////11X,5("* "),"REG/PI DESIGN",5(" *")////)
      NSIZE=NRD*(5*NRD+2*NND)+NNPR*NNPR
      IF(NSIZE.LE.NVRPI) GO TO 5
      WRITE 101,NSIZE
101   FORMAT("0INSUFFICIENT MEMORY /CREGPI/, NEED: ",I4)
      GO TO 8
5     NSIZE=NNPR*(3*NNPR+NRD)
      IF(NSIZE.LE.NVSM) GO TO 10
      WRITE 102,NSIZE
102   FORMAT("0INSUFFICIENT MEMORY /SYSMTX/, NEED: ",I4)
8     LABORT=NSIZE
      RETURN
10    LX=1
      LU=LX+NNPR*NNPR
      CALL WXUS(SM(LX),SM(LU),COM1,ZM1,ZM2,IMPLIC)
      LUIST=LU+NNPR*NRD
      LPHP=LUIST+NNPR*NNPR
      CALL PXUP(CTL(LPHDL),CTL(LBDL),SM(LX),SM(LU),COM1,ZM2,
     1 SM(LUIST),SM(LPHP),SM(LX),ZM1)
      CALL DRIC(NDIM,SM(LPHP),ZM2,SM(LX),ZM1,RPI(LPHCL))
      CALL GCSTAR(SM(LPHP),CTL(LBDL),SM(LU),ZM1,SM(LUIST),SM(LX),ZM2)
      CALL TFRMTX(RPI(LGC1),SM(LX),NRD,NND,1)
      L3=LADDR(NDIM,1,NND+1)+LX-1
      CALL TFRMTX(RPI(LGC2),SM(L3),NRD,NRD,1)
      L1=LADDR(NNPR,NND+1,NND+1)
      L2=LADDR(NNPR,1,NND+1)
      CALL GMINV(NRD,NRD,ZM1(L1),ZM1(L2),MR,1)
      L1=LADDR(NNPR,NND+1,1)
      CALL MMUL(ZM1(L2),ZM1(L1),NRD,NRD,NND,ZM1)
      CALL MMUL(SM(L3),ZM1,NRD,NRD,NND,ZM1(L1))
      CALL MADD1(NRD,NND,SM(LX),ZM1(L1),ZM1,-1.)
      CALL TFRMTX(RPI(LEL),ZM1,NRD,NND,1)
      CALL FMMUL(RPI(LEL),CTL(LPI12),NRD,NND,NRD,RPI(LEE))
      CALL FMADD(RPI(LEE),CTL(LPI22),NRD,NRD,RPI(LEE))
      CALL FTMTX(RPI(LEE),RPI(LEL),NRD,NRD)
      CALL MATLST(RPI(LGC1),NRD,NND,"GC1",KLIST)
      CALL MATLST(RPI(LGC1),NRD,NND,"GC1",KTERM)
      CALL MATLST(RPI(LGC2),NRD,NRD,"GC2",KLIST)
      CALL MATLST(RPI(LGC2),NRD,NRD,"GC2",KTERM)
      CALL MATLST(RPI(LEE),NRD,NRD,"E",KLIST)
      IF(NODY.EQ.1)GO TO 15
      CALL FMMUL(RPI(LEE),DM(LDY),NRD,NRD,NRD,ZM1)
      CALL SUBI(ZM1,NRD,NRD)
      NR2=NRD*NRD
      DO 12 I=1,NR2
   12 ZM1(I)=-ZM1(I)
      NDIM=NRD
      NDIM1=NDIM+1
      CALL GMINV(NRD,NRD,ZM1,ZM2,MR,1)
      CALL FMMUL(ZM2,RPI(LEE),NRD,NRD,NRD,ZM1)
      CALL MATLST(ZM1,NRD,NRD,"KP",KLIST)
      CALL FMMUL(ZM2,RPI(LGC2),NRD,NRD,NRD,ZM1)
      CALL MATLST(ZM1,NRD,NRD,"KI",KLIST)
      GO TO 20
   15 WRITE(KLIST,103)
  103 FORMAT("OKP = E  AND  KI = GC2")
   20 CONTINUE
      LFLRPI=1
      LFLCGT=0
      RETURN
C END SUBROUTINE SREGPI


      SUBROUTINE SCGT
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/ZMTX1/NVZM,ZM1(1)
      COMMON/ZMTX2/ZM2(1)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/NDIMC/NNC,NRC,NPC
      COMMON/CMDMTX/NVCM,NEWCM,NODC,CM(1)
      COMMON/LREGPI/LXDW,LUDW,LPHCL,LGC1,LGC2,LEL,LEE
      COMMON/CREGPI/NVRPI,RPI(1)
      COMMON/LCGT/LA11,LA13,LA21,LA23,LA12,LA22,LELA11,LELA12,LELA13
      COMMON/CCGT/NVCGT,CGT(1)
      IF(NEWCM) 20,20,15
   15 NSIZE=(NND+2*NPD)*(NNC+NRC+NDD)
      IF(NSIZE.LE.NVCGT) GO TO 16
      WRITE 106,NSIZE
      LABORT=NSIZE
      RETURN
   16 IF(NND.GE.NNC) GO TO 17
      WRITE 107
      GO TO 18
   17 IF(NND.GE.NDD) GO TO 19
      WRITE 108
   18 LABORT=-1
      RETURN
   19 NEWCM=0
      LA11=1
      LA13=LA11+NND*NNC
      LA21=LA13+NND*NDD
      LA23=LA21+NPD*NNC
      LA12=LA23+NPD*NDD
      LA22=LA12+NND*NRC
      LELA11=LA22+NPD*NRC
      LELA12=LELA11+NPD*NNC
      LELA13=LELA12+NPD*NRC
      CALL CGTA(CGT(LA11),CGT(LA13),CGT(LA21),CGT(LA23),CGT(LA12),
     1 CGT(LA22),ZM1,ZM2)
   20 CALL CGTKX(CGT(LA11),CGT(LA13),CGT(LA21),CGT(LA23),CGT(LA12),
     1 CGT(LA22),CGT(LELA11),CGT(LELA12),CGT(LELA13),RPI(LGC1))
      LFLCGT=1
  106 FORMAT("0INSUFFICIENT MEMORY /CCGT/, NEED: ",I4)
  107 FORMAT("0FEWER DESIGN MODEL THAN COMMAND MODEL STATES")
  108 FORMAT("FEWER DESIGN MODEL THAN DISTURBANCE MODEL STATES")
      RETURN
C END SUBROUTINE SCGT
      END


      SUBROUTINE CGTKX(A11,A13,A21,A23,A12,A22,RELA11,RELA12,RELA13,RGC)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/NDIMC/NNC,NRC,NPC
      DIMENSION A11(1),A13(1),A21(1),A23(1),A12(1),A22(1),
     1 RELA11(1),RELA12(1),RELA13(1),RGC(1)
      NDIM=NRD
      NDIM1=NDIM+1
      CALL FMMUL(RGC,A11,NRD,NND,NNC,RELA11)
      CALL MADD1(NRD,NNC,RELA11,A21,RELA11,1.)
      CALL MATLST(RELA11,NRD,NNC,"KXM",KLIST)
      CALL MATLST(RELA11,NRD,NNC,"KXM",KTERM)
      CALL FMMUL(RGC,A12,NRD,NND,NRC,RELA12)
      CALL MADD1(NRD,NRC,RELA12,A22,RELA12,1.)
      CALL MATLST(RELA12,NRD,NRC,"KXU",KLIST)
      CALL MATLST(RELA12,NRD,NRC,"KXU",KTERM)
      IF(NDD.LT.1) RETURN
      CALL FMMUL(RGC,A13,NRD,NND,NDD,RELA13)
      CALL MADD1(NRD,NDD,RELA13,A23,RELA13,1.)
      CALL MATLST(RELA13,NRD,NDD,"KXN",KLIST)
      CALL MATLST(RELA13,NRD,NDD,"KXN",KTERM)
      RETURN
C END SUBROUTINE CGTKX
      END


      SUBROUTINE CTRESP(VX0,VX1,X0,X1,XM0,XM1,ZM1,NVOUT,TEND,IUM,VUM,
     1 NST)
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/LOCD/LAP,LGP,LPHI,LBD,LEX,LPHD,LQ,LQN,LQD,LC,LDY,LEY,LHP,LR
      COMMON/DSNMTX/NVDM,NODY,NOEY,DM(1)
      COMMON/NDIMC/NNC,NRC,NPC
      COMMON/LOCC/LPHC,LBDC,LCC,LDC
      COMMON/CMDMTX/NVCM,NEWCM,NODC,CM(1)
      COMMON/LOCT/LPHT,LBDT,LQDT,LHT,LRT,LTDT,LTNT
      COMMON/TRUMTX/NVTM,TM(1)
      COMMON/LREGPI/LXDW,LUDW,LPHCL,LGC1,LGC2,LEL,LEE
      COMMON/CREGPI/NVRPI,RPI(1)
      DIMENSION VX0(1),VX1(1),X0(1),X1(1),XM0(1),XM1(1),ZM1(1)
      NSTPO=.01*TEND/TSAMP+.5
      NST=2
      IF(NSTPO.GE.1) GO TO 1
      NSTPO=1
      NST=1
 1    NSTEPS=100*NSTPO
      IF(LFLCGT.EQ.0) GO TO 2
      LMO=NVOUT-NPC
      IF(NDD.EQ.0) GO TO 4
      LDCGT=1
      GO TO 5
 2    LMO=NVOUT
 4    LDCGT=0
 5    LU=LMO-NRD
      LSO=LU-NPD
      NVX=LSO-1
      IF(LTEVAL)6,6,10
 6    DO 7 I=1,NVX
 7    X1(I)=VX1(I)
      GO TO 12
 10   CALL XFDT(VX1,X1,LDCGT)
 12   NNDP1=NND+1
      REWIND KPLOT
      CALL YDSN(X1,VX1(LU),DM(LC),DM(LDY),LDCGT,VX1(LSO))
      IF(LFLCGT.EQ.1) CALL YCMD(XM1,IUM,VUM,CM(LCC),CM(LDC),
     1 VX1(LMO))
      CALL WPLOTF(VX1,NVOUT)
      DO 100 IT=1,NSTEPS
      CALL URPI(RPI(LGC1),RPI(LGC2),DM(LC),DM(LDY),X0,X1,VX0(LU),
     1 VX1(LU))
      IF(LFLCGT) 20,20,15
 15   CALL UCGT(VX0(LU),VX1(LU),XM0,XM1,X0(NNDP1),ZM1,IUM,VUM,IT,
     1 VX1(LMO))
      CALL CUPDAT(XM0,XM1,IUM,VUM)
 20   CALL FTMTX(VX1(LU),VX0(LU),NRD,1)
      CALL FTMTX(X1,X0,NPLD,1)
      IF(LTEVAL) 25,25,30
 25   CALL DUPDAT(DM(LPHI),DM(LBD),DM(LPHD),DM(LEX),X0,X1,
     1 VX1,VX0(LU),LDCGT,NNDP1)
      GO TO 35
 30   CALL TUPDAT(TM(LPHT),TM(LBDT),VX0,VX1,VX0(LU))
      CALL XFDT(VX1,X1,LDCGT)
 35   IF(MOD(IT,NSTPO).NE.0) GO TO 100
      VX1(NVOUT)=TSAMP*FLOAT(IT)
      CALL YDSN(X1,VX1(LU),DM(LC),DM(LDY),LDCGT,VX1(LSO))
      IF(LFLCGT.EQ.1) CALL YCMD(XM1,IUM,VUM,CM(LCC),CM(LDC),
     1 VX1(LMO))
      CALL WPLOTF(VX1,NVOUT)
 100  CONTINUE
```

```
      ENDFILE KPLOT
      RETURN
C END SUBROUTINE CTRESP
      END
```

```
      SUBROUTINE UCGT(U0,U1,XM0,XM1,DDIF,ZM1,IUM,VUM,IT,YC)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/NDIMC/NNC,NRC,NPC
      COMMON/LOCC/LPHC,LBDC,LCC,LDC
      COMMON/CMDMTX/NVCM,NEWCM,NODC,CM(1)
      COMMON/LREGPI/LXDW,LUDW,LPHCL,LGC1,LGC2,LEL,LEE
      COMMON/CREGPI/NVRPI,RPI(1)
      COMMON/LCGT/LA11,LA13,LA21,LA23,LA12,LA22,LELA11,LELA12,LELA13
      COMMON/CCGT/NVCGT,CGT(1)
      DIMENSION U0(1),U1(1),XM0(1),XM1(1),DDIF(1),ZM1(1)
      CALL YCMD(XM0,IUM,VUM,CM(LCC),CM(LDC),U0)
      IF(IT.GT.1) GO TO 8
      I=LELA12+LADDR(NPD,1,IUM)-1
      CALL MADD1(NPD,1,U1,CGT(I),U1,VUM)
      CALL MMULS(RPI(LEL),U0,NDIM,NDIM,1,U1)
      GO TO 10
8     CALL MADD1(NPD,1,YC,U0,YC,-1.)
      CALL MMULS(RPI(LEL),YC,NDIM,NDIM,1,U1)
10    CALL MMULS(RPI(LGC2),U0,NDIM,NDIM,1,U1)
      DO 12 I=1,NNC
12    XM0(I)=XM1(I)-XM0(I)
      CALL FMMUL(CGT(LELA11),XM0,NPD,NNC,1,U0)
      CALL VADD(NDIM,1.,U1,U0)
      IF(NDD.EQ.0) RETURN
      DO 14 I=1,NDD
14    DDIF(I)=-DDIF(I)
      CALL MMULS(CGT(LELA13),DDIF,NPD,NDD,1,U1)
      RETURN
C END SUBROUTINE UCGT
      END
```

C.4 Subroutine Changes for CGT/PI Formulation 4

Subroutines SCGT and CGTKX were changed to produce the program based on CGT/PI formulation 4. These subroutines are listed on the following pages. Changed lines are marked.

```
      SUBROUTINE SCGT
      COMMON/DESIGN/NVCOM,TSAMP,LFLRPI,LFLCGT,LFLKF,LTEVAL,LABORT
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/ZMTX1/NVZM,ZM1(1)
      COMMON/ZMTX2/ZM2(1)
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/NDIMC/NNC,NRC,NPC
      COMMON/CMDMTX/NVCM,NEWCM,NODC,CM(1)
      COMMON/LREGPI/LXDW,LUDW,LPHCL,LGC1,LGC2,LEL,LEE
      COMMON/CREGPI/NVRPI,RPI(1)
      COMMON/LCGT/LA11,LA13,LA21,LA23,LA12,LA22,LELA11,LELA12,LELA13
      COMMON/CCGT/NVCGT,CGT(1)
      IF(NEWCM) 20,20,15
   15 NSIZE=(NND+2*NPD)*(NNC+NRC+NDD)
      IF(NSIZE.LE.NVCGT) GO TO 16
      WRITE 106,NSIZE
      LABORT=NSIZE
      RETURN
   16 IF(NND.GE.NNC) GO TO 17
      WRITE 107
      GO TO 18
   17 IF(NND.GE.NDD) GO TO 19
      WRITE 108
   18 LABORT=-1
      RETURN
   19 NEWCM=0
      LA11=1
      LA13=LA11+NND*NNC
      LA21=LA13+NND*NDD
      LA23=LA21+NPD*NNC
      LA12=LA23+NPD*NDD
      LA22=LA12+NND*NRC
      LELA11=LA22+NPD*NRC
      LELA12=LELA11+NPD*NNC
      LELA13=LELA12+NPD*NRC
      CALL CGTA(CGT(LA11),CGT(LA13),CGT(LA21),CGT(LA23),CGT(LA12),
     1 CGT(LA22),ZM1,ZM2)
   20 CALL CGTKX(CGT(LA11),CGT(LA13),CGT(LA21),CGT(LA23),CGT(LA12),
     1 CGT(LA22),CGT(LELA11),CGT(LELA12),CGT(LELA13),RPI(LGC1))
      LFLCGT=1
  106 FORMAT("0INSUFFICIENT MEMORY /CCGT/, NEED: ",I4)
  107 FORMAT("0FEWER DESIGN MODEL THAN COMMAND MODEL STATES")
  108 FORMAT("FEWER DESIGN MODEL THAN DISTURBANCE MODEL STATES")
      RETURN
C END SUBROUTINE SCGT
      END


      SUBROUTINE CGTKX(A11,A13,A21,A23,A12,A22,RELA11,RELA12,RELA13,RGC)
      COMMON/MAIN1/NDIM,NDIM1,COM1(1)
      COMMON/FILES/KSAVE,KDATA,KPLOT,KLIST,KTERM
      COMMON/NDIMD/NND,NRD,NPD,NMD,NDD,NWD,NWDD,NPLD,NWPNWD,NNPR
      COMMON/NDIMC/NNC,NRC,NPC
      DIMENSION A11(1),A13(1),A21(1),A23(1),A12(1),A22(1),
     1 RELA11(1),RELA12(1),RELA13(1),RGC(1)
      NDIM=NRD
      NDIM1=NDIM+1
      CALL FMMUL(RGC,A11,NRD,NND,NNC,RELA11)
      CALL MADD1(NRD,NNC,RELA11,A21,RELA11,1.)
      CALL MATLST(RELA11,NRD,NNC,"KXM",KLIST)
      CALL MATLST(RELA11,NRD,NNC,"KXM",KTERM)
      CALL FMMUL(RGC,A12,NRD,NND,NRC,RELA12)
      CALL MADD1(NRD,NRC,RELA12,A22,RELA12,1.)
      CALL MATLST(RELA12,NRD,NRC,"KXU",KLIST)
      CALL MATLST(RELA12,NRD,NRC,"KXU",KTERM)
      IF(NDD.LT.1) RETURN
      CALL FMMUL(RGC,A13,NRD,NND,NDD,RELA13)
      CALL MADD1(NRD,NDD,RELA13,A23,RELA13,1.)
      CALL MATLST(RELA13,NRD,NDD,"KXN",KLIST)
      CALL MATLST(RELA13,NRD,NDD,"KXN",KTERM)
      RETURN
C END SUBROUTINE CGTKX
      END
```

Appendix D

CGT/PI/KF Design Evaluation

D.1 Introduction

CGT/PI/KF design actually consists of three separate designs: the controller with PI control action, the open-loop CGT or closed-loop CGT/PI controller, and the Kalman filter. Each of these designs is best evaluated according to criteria specifically related to the task that each is to perform.

For the controller, relevant considerations include the closed-loop system poles, the values of the feedforward and/or feedback gains, and the time response of the controller system states, output, and control inputs in either unforced or forced input conditions. For the filter, relevant considerations include the poles of the filter, the values of the filter gains, and the filter's estimation error behavior. An overall evaluation of the system performance for the CGT/PI/KF closed -loop controller is necessary to tune the entire design properly and judge its ultimate performance. The true CGT/PI/KF controller will suffer some degree of degraded performance due to the dynamics of the Kalman filter's state estimation and also a slightly increased delay in control generation due to the needed filter computations. The necessary performance evaluation software for this entire CGT/PI/KF controller is contained in the program "PFEVAL" (Ref 21). This appendix is taken from Chapter IV of Floyd's thesis (Ref 9).

D.2 PI Controller Evaluation

The discrete-time poles of the closed-loop system incorporating the optimal gain $\underline{G}_c^*$ of Equation (III-52) and assuming perfect state knowledge are computed from the matrix

$$\underline{\Phi}_{\delta CL} = [\underline{\Phi}_{\delta} - \underline{B}_{\delta}\underline{G}_c^*] \tag{D-1}$$

with $\underline{\Phi}_{\delta}$ and $\underline{B}_{\delta}$ as defined in Equations (III-42a) and (III-42b), respectively. The equivalent continuous-time poles are then computed using the inverse of the relation between the z and s transforms

$$z = e^{sT} \tag{D-2}$$

in which z and s are respectively the discrete and continuous-time complex poles and T is the controller sample period. This mapping is for the primary strip in the s-plane only and does not consider any possible aliasing effects (Ref 14).

Another important consideration in evaluating the controller design is the magnitudes of the gains $\underline{G}_{c1}^*$ and $\underline{G}_{c2}^*$, but the most useful information can be determined from the system's time response to initial conditions on the states. The time response can be readily simulated using the discretized deterministic state transition equations of either the design or the truth model and driven by the control input

$$\underline{u}^*(t_i) = \underline{u}^*(t_{i-1}) - \underline{G}_{c1}^*[\underline{x}(t_i) - \underline{x}(t_{i-1})]$$
$$- \underline{G}_{c2}^*[\underline{C} \quad \underline{D}_y] \begin{bmatrix} \underline{x}(t_{i-1}) \\ \underline{u}^*(t_{i-1}) \end{bmatrix} \qquad \text{(D-3)}$$

obtained from Equation (III-75) by deleting terms due to the CGT feedforward control. Note that this controller form is valid only for the PI controller with zero reference input ($\underline{y}_d = \underline{0}$) and in response to non-zero initial conditions on the states. Plots of the time histories of the states, the outputs, and the generated control inputs then may be made. These allow evaluation of the quality of the regulation achieved--speed and damping of the state and output response, and the magnitudes and rates of the control inputs actually required.

D.3 CGT or CGT/PI Evaluation

As for the PI controller evaluation discussed above, consideration of the magnitudes of the feedforward gains may provide useful insights into the CGT design results. For the open-loop CGT the relevant gains are $\underline{A}_{21}$ acting on the command model states and $\underline{A}_{23}$ acting on the disturbance states in the open-loop CGT control law

$$\underline{u}(t_i) = \underline{u}(t_{i-1}) + \underline{A}_{21}[\underline{x}_m(t_i) - \underline{x}_m(t_{i-1})] + \underline{A}_{22}[\underline{u}_m(t_i) - \underline{u}_m(t_{i-1})] + \underline{A}_{23}[\underline{n}_d(t_i) - \underline{n}_d(t_{i-1})] \quad \text{(D-4)}$$

obtained from Equation (III-75) by setting the PI controller gains $\underline{G}_{c1}^*$ and $\underline{G}_{c2}^*$ to zero. For the CGT/PI controller of Equation (III-75) the relevant gains are

$$\underline{K}_{xm} = \underline{L}\underline{A}_{11} + \underline{A}_{21} \quad \text{(D-5a)}$$

$$\underline{K}_{xu} = \underline{L}\underline{A}_{12} + \underline{A}_{22} \quad \text{(D-5b)}$$

$$\underline{K}_{xn} = \underline{L}\underline{A}_{13} + \underline{A}_{23} \quad \text{(D-5c)}$$

acting on the command model states and inputs, and disturbance states, respectively. (See Section 3.3.2 for changes for the CGT/PI formulations 2,3,4.)

D.4 Kalman Filter Evaluation

Without going into any detail, the state transition matrix for the filter estimate propagation is

$$\underline{\Phi}_{KF} = [\underline{I} - \underline{K}_F\underline{H}_a]\underline{\Phi}_a \quad \text{(D-6)}$$

The filter's discrete-time poles are computed as the eigenvalues of $\underline{\Phi}_{KF}$ and their continuous-time equivalents computed by the method mentioned in Section D.2. The magnitudes of the Kalman filter gains may be evaluated from the standard $\underline{K}_F$ matrix

$$\underline{K}_F = \underline{P}_a\underline{H}_a^T[\underline{H}_a\underline{P}_a\underline{H}_a^T + \underline{R}]^{-1} \tag{D-7}$$

The greatest insight into Kalman filter tuning is provided by a covariance analysis (Refs 11 and 17). In the covariance analysis, the covariance of the estimation errors of the Kalman filter when applied to the system truth model is propagated forward in time from the initial conditions on the covariances of the truth model states. In parallel with this estimate-error covariance propagation, the filter-computed covariance is itself propagated forward in time. The true and filter computed estimation error covariances may then be compared since the truth model state estimation error covariances can be transformed to errors for the design states using Equation (B-6c). The designer may then modify (tune) the dynamics noise and measurement noise strengths to achieve the desired filter performance: the duration of the initial estimation transient and the steady state error covariance obtained. The total development of this is given in Ref (17) and summarized in Ref (9).

Vita

James Page McMillian was born on 2 February 1947 in Clinton, Missouri. He enlisted in the Air Force in June 1970 and received training as an Electronics Technician for the Precision Measuring Equipment Laboratories. Assignments included England AFB, La., Lowry AFB, Colo., Barksdale AFB, La., and Bitburg AFB, Germany. In September 1977 he attended Louisiana Tech University under the Airman Education and Commissioning Program, graduating in December 1977 with a degree of Bachelor of Science in Electrical Engineering. During his undergraduate education, he became a member of Eta Kappa Nu and Tau Beta Pi. He received his commission from Officer Training School in March 1980. From March 1980 until June 1982 he was assigned to HQ Electronic Security Command, Kelly AFB, San Antonio, Texas. He entered the School of Engineering of the Air Force Institute of Technology in June 1982 and has pursued the Guidance and Control curriculum of the Electrical Engineering Department.

Permanent Address: Rt. 1 Box 28A

Calhoun, Missouri 65323

8a. NAME OF FUNDING/SPONSORING ORGANIZATION	**8b. OFFICE SYMBOL** *(If applicable)*	**9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER**

8c. ADDRESS *(City, State and ZIP Code)*	**10. SOURCE OF FUNDING NOS.**			
	PROGRAM ELEMENT NO.	**PROJECT NO.**	**TASK NO.**	**WORK UNI NO.**
11. TITLE *(Include Security Classification)* See Box 19				

12. PERSONAL AUTHOR(S)
James P. McMillian, B.S., B.S.E.E., 1Lt, USAF

13a. TYPE OF REPORT	**13b. TIME COVERED**	**14. DATE OF REPORT** *(Yr., Mo., Day)*	**15. PAGE COUNT**
MS Thesis	FROM ________ TO ________	1983 December	356

16. SUPPLEMENTARY NOTATION

Approved for public release: IAW AFR 190-17.
[illegible] 7 Feb 84
LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB OH 45433

17. COSATI CODES		
FIELD	**GROUP**	**SUB. GR.**
01	03	

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*

Optimal Control, Model-Following Control,
Proportional-Integral Control,
Command Generator Tracker

This study develops a computer program for interactive execution to aid in the design of Command Generator Tracker control systems employing Proportional Plus Integral inner loop controllers based on the integral of the regulation error and Kalman Filters for state estimation (CGT/PI/KF controllers). Sampled-data controller designs are based upon the Linear system model, Quadratic cost, and Gaussian noise process (LQG) assumptions of optimal control theory.

The report develops the CGT/PI/KF controller theory with the PI controller portion based upon the integral of the regulation error. Following a brief description of the computer program developed, results of applying it to an example aircraft-related controller design problem is presented. The CGT/PI/KF controller is found to be a technique particularly well suited to the aircraft control design problem but is a technique that may be applied to any general controller problem fitting the design criteria.

Use of the computer program is fully documented in the appendices of the report. Included are a brief "Programmer's Manual," a complete "User's Manual", and a program listing. These pertain to the computer program as implemented on a CDC CYBER computer system.

FILMED